

Progress Application Development Environment — Getting Started

© 2002 Progress Software Corporation. All rights reserved.

Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. This manual is also copyrighted and all rights are reserved. This manual may not, in whole or in part, be copied, photocopied, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Progress Software Corporation.

The information in this manual is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear in this document.

The references in this manual to specific platforms supported are subject to change.

Progress, Powered by Progress, Progress Fast Track, Progress Profiles, Partners in Progress, Partners en Progress, Progress en Partners, Progress in Progress, P.I.P., Progress Results, ProVision, ProCare, ProtoSpeed, SmartBeans, SpeedScript, and WebSpeed are registered trademarks of Progress Software Corporation in the U.S. and other countries. A Data Center of Your Very Own, Allegrix, Apptivity, AppsAlive, AppServer, ASPen, ASP-in-a-Box, Empowerment Center, Fathom, Future Proof, IntelliStream, OpenEdge, PeerDirect, POSSE, POSSENET, Progress Dynamics, Progress Software Developers Network, SectorAlliance, SmartObjects and WebClient are trademarks or service marks of Progress Software Corporation in the U.S. and other countries.

SonicMQ is a registered trademark of Sonic Software Corporation in the U.S. and other countries.

Vermont Views is a registered trademark of Vermont Creative Software in the U.S. and other countries.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Any other trademarks and/or service marks contained herein are the property of their respective owners.

Progress Software Corporation acknowledges the use of Raster Imaging Technology copyrighted by Snowbound Software 1993-1997, the IBM XML Parser for Java Edition, and software developed by the Apache Software Foundation (http://www.apache.org/).

© IBM Corporation 1998-1999. All rights reserved. U.S. Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Progress is a registered trademark of Progress Software Corporation and is used by IBM Corporation in the mark Progress/400 under license. Progress/400 AND 400® are trademarks of IBM Corporation and are used by Progress Software Corporation under license.

The implementation of the MD5 Message-Digest Algorithm used by Progress Software Corporation in certain of its products is derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

May 2002



Product Code: 4503
Item Number: 91586;V9.1D

Contents

Preface . . . . . **ix**
Purpose ix
Audience ix
Organization Of This Manual ix
How To Use This Manual x
Typographical Conventions xi
Progress Messages xii
Other Useful Documentation xiii
Getting Started xiii
Development Tools xiv
Reporting Tools xvi
4GL xvi
Database xvii
DataServers xviii
SQL-89/Open Access xviii
SQL-92 xix
Deployment xx
WebSpeed xx
Reference xxi

1. Overview Of the Progress Application Development Environment **1–1**
1.1 Starting the ADE 1–2
1.1.1 Starting a Tool From the ADE Desktop 1–2
1.1.2 Starting a Tool From Another Tool 1–3
1.2 ADE Toolset 1–3
1.3 Progress Online Help System 1–5
1.4 Overview Of the AppBuilder 1–6
1.4.1 Starting the AppBuilder 1–6
1.4.2 AppBuilder Main Window 1–7

1.4.3 AppBuilder Object Palette 1–11
1.4.4 Advanced Editing Features For Progress Editors 1–17

2. Working With the Sample Application 2–1
2.1 The SportsPro Sample Application 2–2
2.2 Creating and Connecting To the Sports2000 Database 2–2
2.3 Copying the Sample Files 2–6
2.4 Compiling the Sample Files 2–6
2.5 Running the Sample Application 2–6
2.5.1 Dynamic Query Module 2–10
2.5.2 Dynamic Buffer/Temp Table Module 2–12
2.5.3 Drag and Drop Module 2–15
2.5.4 Resizable Dynamic Browser 2–17
2.5.5 SmartDataObject Module 2–20
2.5.6 SmartDataField Module 2–22
2.5.7 Publish and Subscribe Module 2–24
2.5.8 SmartB2B and SonicMQ Module 2–27
2.6 Working With the Order Tracking System 2–30
2.6.1 Customer Section 2–34
2.6.2 Order Section 2–39
2.6.3 Order Line Section 2–47
2.6.4 Menu Items 2–49

3. Assembling the Order Tracking System 3–1
3.1 Progress Advisor 3–2
3.2 Tutorial Structure 3–2
3.3 Assembling the Main Window 3–3
3.3.1 Task List and Short Procedure 3–3
3.3.2 Detailed Main Window Procedure 3–5
3.4 Building the Customer Section 3–22
3.4.1 Task List and Short Procedure 3–22
3.4.2 Detailed Customer Section Procedure 3–24
3.5 Building the Order Section 3–39
3.5.1 Task List and Short Procedure 3–40
3.5.2 Detailed Order Section Procedure 3–43
3.6 Building the Order Line Section 3–63
3.6.1 Task List and Short Procedure 3–64
3.6.2 Detailed Order Line Section Procedure 3–66

4. An Overview Of SmartObject Technology 4–1
4.1 SmartObject Technology 4–2
4.2 SmartObjects Used In the Order Tracking System 4–2
4.2.1 Naming Conventions 4–8
4.3 Business Scenario 4–9

4.4 Development Phases 4–10
4.4.1 Analysis and Design Phase—What SmartObjects Are Needed? 4–11
4.4.2 Building Phase—Creating the SmartObject Masters 4–11
4.4.3 Assembly Phase 4–11
4.5 Building SmartDataObjects 4–12
4.5.1 Starting the SmartDataObject Wizard 4–13
4.5.2 Defining the Query 4–14
4.5.3 Saving the New SmartDataObject 4–19
4.6 Building SmartDataViewers 4–21
4.6.1 Starting the SmartDataViewer Wizard 4–22
4.6.2 Selecting the Fields to Display 4–25
4.6.3 Arranging Fields In the SmartDataViewer 4–29
4.6.4 Adding and Modifying Field Labels 4–31
4.6.5 Disabling Fields 4–32
4.6.6 Saving the New SmartDataViewer 4–33
4.7 Building SmartDataBrowsers 4–35
4.7.1 Starting the SmartDataBrowser Wizard 4–36
4.7.2 Selecting Fields for the SmartDataBrowser 4–38
4.7.3 Resizing the SmartDataBrowser 4–42
4.7.4 Saving the New SmartDataBrowser 4–44

5. Where To Go From Here 5–1
5.1 The Progress Documentation Set 5–3
5.2 Education Services 5–3
5.3 Hands-on Training 5–4
5.4 Consulting Services 5–4
5.4.1 Phone-in Consultation 5–5
5.5 Technical Services 5–5
5.5.1 Contacting Technical Services 5–6
5.5.2 Progress Technical Support Centers 5–6
5.5.3 TechWeb 5–7
5.5.4 Progress Knowledge Base 5–7

Index Index–1

Figures

Figure 1–1: ADE Desktop 1–2
Figure 1–2: AppBuilder Main Window and Object Palette—Initial Display 1–6
Figure 1–3: The AppBuilder Main Window 1–7
Figure 1–4: The AppBuilder Object Palette 1–12
Figure 4–1: SmartObjects In the Order Tracking System 4–3
Figure 4–2: SmartObject Types and SmartLinks 4–5
Figure 4–3: SmartFolder and Page SmartLink 4–6
Figure 4–4: SmartLinks For dcust SDO 4–7
Figure 4–5: SmartLinks For dorder SDO 4–7
Figure 4–6: SmartLinks For dordlne SDO 4–8

Tables

Table 1–1:	Toolbar Icons	1–8
Table 1–2:	Object Palette Icons	1–12
Table 2–1:	Data Dictionary Icons	2–5
Table 2–2:	Components Of the SportsPro Main Window	2–8
Table 2–3:	Components Of the Dynamic Query Module	2–11
Table 2–4:	Components Of the Dynamic Buffer/Temp Table Module	2–14
Table 2–5:	Components Of the Drag and Drop Module	2–17
Table 2–6:	Components Of the Resizable Dynamic Browser Module	2–19
Table 2–7:	Components Of the SmartDataObject Module	2–21
Table 2–8:	Components Of the SmartDataField Module	2–23
Table 2–9:	Components Of the Publish and Subscribe Module	2–25
Table 2–10:	Components Of the SmartB2B and SonicMQ Module	2–28
Table 2–11:	Components Of the Order Tracking System Main Window	2–32
Table 2–12:	Components Of the Maintain Customer Section	2–38
Table 2–13:	Components Of the Order Entry/Maintenance Section	2–42
Table 2–14:	Components Of the Find Order Section	2–46
Table 2–15:	Components Of the Inventory Transfer Section	2–50
Table 2–16:	Components Of the Set Application Language Section	2–52
Table 4–1:	AppBuilder Filename Conventions	4–9

Preface

Purpose

This guide introduces you to the Progress technology and tools that you can use to easily develop and maintain applications. After a brief introduction to the Application Development Environment (ADE) Desktop and toolset, you can complete tutorials and exercises that show you how to use the Progress AppBuilder to easily create and maintain component-based applications using the Progress SmartObject technology.

Audience

This guide is for anyone who creates and maintains end-user applications using the Progress Application Development Environment (ADE). This guide contains valuable information for experienced Progress developers as well as developers who are new to Progress and the ADE.

Organization Of This Manual

This guide provides an introduction and overview of the ADE Desktop and its tools, and the Progress SmartObject technology. In addition to conceptual and overview information, the guide includes tutorials and exercises that you can complete to better understand the ADE tools and the Progress technology.

Chapter 1, “Overview Of the Progress Application Development Environment”

Provides an overview of the ADE Desktop and introduces you to the tools that comprise the ADE. In addition to a general description of the ADE, this chapter provides a more detailed description of the AppBuilder.

Chapter 2, “Working With the Sample Application”

Provides information about how to run and use the SportsPro sample application. By working with the sample application, you can better understand the AppBuilder and the Progress technology. This chapter also details the structure of each module in the sample application.

Chapter 3, “Assembling the Order Tracking System”

Provides exercises for assembling the main window of the SportsPro Order Tracking System from a set of SmartObjects that Progress supplies. Assembling the main window provides an understanding of how to work with Progress SmartObjects.

Chapter 4, “An Overview Of SmartObject Technology”

Provides an introduction to SmartObject technology and conceptual information about the different phases of designing and building an application. This chapter also includes exercises that you complete to better understand how to use the AppBuilder and how to fabricate SmartObject masters for use in building applications.

Chapter 5, “Where To Go From Here”

Describes other manuals in the Progress documentation set and provides information about how to take advantage of the educational, consulting, and technical support resources of Progress Software Corporation.

How To Use This Manual

This guide assumes that you have successfully installed Progress as described in the *Progress Installation and Configuration Guide Version 9 for Windows*, and that you have created a working directory for the files you create as you work through the tutorials and exercises. The examples in this book use `\Progress\Wrk` as the working directory.

The material has been chosen and organized so that you can build upon your knowledge of the ADE, AppBuilder, and the Progress SmartObject technology. The presentation of information has been designed so that:

- The terms, concepts, and techniques most fundamental to a practical understanding of the ADE and the AppBuilder are introduced first.
- The opportunity to create a database and database connection and to interact with an application that was built using Progress tools and technology is presented next.

- An opportunity to assemble part of the sample application using the AppBuilder and prebuilt SmartObjects is presented after you have worked with the sample application.
- An overview of SmartObject technology and exercises for fabricating objects are presented after you have had an opportunity to use the ADE to build part of the sample application.

Typographical Conventions

This manual uses the following typographical conventions:

- **Bold typeface** indicates:
  - Commands or characters that the user types
  - That a word carries particular weight or emphasis
- *Italic typeface* indicates:
  - Progress variable information that the user supplies
  - New terms
  - Titles of complete publications
- `Monospaced typeface` indicates:
  - Code examples
  - System output
  - Operating system filenames and pathnames

The following typographical conventions are used to represent keystrokes:

- Small capitals are used for Progress key functions and generic keyboard keys.

  END-ERROR, GET, GO
  ALT, CTRL, SPACEBAR, TAB

- When you have to press a combination of keys, they are joined by a dash. You press and hold down the first key, then press the second key.

  CTRL-X

- When you have to press and release one key, then press another key, the key names are separated with a space.

  ESCAPE H
  ESCAPE CURSOR-LEFT

Progress Messages

Progress displays several types of messages to inform you of routine and unusual occurrences:

- Execution messages inform you of errors encountered while Progress is running a procedure (for example, if Progress cannot find a record with a specified index field value).
- Compile messages inform you of errors found while Progress is reading and analyzing a procedure prior to running it (for example, if a procedure references a table name that is not defined in the database).
- Startup messages inform you of unusual conditions detected while Progress is getting ready to execute (for example, if you entered an invalid startup parameter).

After displaying a message, Progress proceeds in one of several ways:

- Continues execution, subject to the error-processing actions that you specify, or that are assumed, as part of the procedure. This is the most common action taken following execution messages.
- Returns to the Progress Procedure Editor so that you can correct an error in a procedure. This is the usual action taken following compiler messages.
- Halts processing of a procedure and returns immediately to the Procedure Editor. This does not happen often.
- Terminates the current session.

Progress messages end with a message number in parentheses. In this example, the message number is 200:

```
** Unknown table name table. (200)
```

Use Progress online help to get more information about Progress messages. Many Progress tools include the following Help menu options to provide information about messages:

- Choose Help→Recent Messages to display detailed descriptions of the most recent Progress message and all other messages returned in the current session.
- Choose Help→Messages, then enter the message number to display a description of any Progress message. (If you encounter an error that terminates Progress, make a note of the message number before restarting.)
- In the Procedure Editor, press the **HELP** key (**F2** or **CTRL-W**).

Other Useful Documentation

This section lists Progress Software Corporation documentation that you might find useful. Unless otherwise specified, these manuals support both Windows and Character platforms and are provided in electronic documentation format on CD-ROM.

Getting Started

Progress Electronic Documentation Installation and Configuration Guide (Hard copy only)

A booklet that describes how to install the Progress EDOC viewer and collection on UNIX and Windows.

Progress Installation and Configuration Guide Version 9 for UNIX

A manual that describes how to install and set up Progress Version 9.1 for the UNIX operating system.

Progress Installation and Configuration Guide Version 9 for Windows

A manual that describes how to install and set up Progress Version 9.1 for all supported Windows and Citrix MetaFrame operating systems.

Progress Version 9 Product Update Bulletin

A bulletin that provides a list of new and changed features by release, a list and description of changes to documentation by release, and critical information about product changes that might require changes to existing code and configurations.

This bulletin also provides information about where to go for detailed information about the new and changed features and documentation.

Progress Language Tutorial for Windows and *Progress Language Tutorial for Character*

Platform-specific tutorials designed for new Progress users. The tutorials use a step-by-step approach to explore the Progress application development environment using the 4GL.

Progress Master Glossary for Windows and *Progress Master Glossary for Character* (EDOC only)

Platform-specific master glossaries for the Progress documentation set. These books are in electronic format only.

Progress Master Index and Glossary for Windows and *Progress Master Index and Glossary for Character* (Hard copy only)

Platform-specific master indexes and glossaries for the Progress hard-copy documentation set.

Progress Startup Command and Parameter Reference

A reference manual that describes the Progress startup and shutdown commands that you use at the command line, and the startup parameters that you use for Progress processes. This guide also provides information about parameter usage and parameter files.

Welcome to Progress (Hard copy only)

A booklet that explains how Progress software and media are packaged. An icon-based map groups the documentation by functionality, providing an overall view of the documentation set. *Welcome to Progress* also provides descriptions of the various services Progress Software Corporation offers.

Development Tools

Progress ADM 2 Guide

A guide to using the Application Development Model, Version 2 (ADM 2) application architecture to develop Progress applications. It includes instructions for building and using Progress SmartObjects.

Progress ADM 2 Reference

A reference for the Application Development Model, Version 2 (ADM 2) application. It includes descriptions of ADM 2 functions and procedures.

Progress AppBuilder Developer's Guide (Windows only)

A programmer's guide to using the Progress AppBuilder visual layout editor. AppBuilder is a Rapid Application Development (RAD) tool that can significantly reduce the time and effort required to create Progress applications.

Progress Basic Database Tools (Character only; information for Windows is in online help)

A guide for the Progress Database Administration tools, such as the Data Dictionary.

Progress Basic Development Tools (Character only; information for Windows is in online help)

A guide for the Progress development toolset, including the Progress Procedure Editor and the Application Compiler.

Progress Debugger Guide

A guide for the Progress Application Debugger. The Debugger helps you trace and correct programming errors by allowing you to monitor and modify procedure execution as it happens.

Progress Help Development Guide (Windows only)

A guide that describes how to develop and integrate an online help system for a Progress application.

Progress Translation Manager Guide (Windows only)

A guide that describes how to use the Progress Translation Manager tool to manage the entire process of translating the text phrases in Progress applications.

Progress Visual Translator Guide (Windows only)

A guide that describes how to use the Progress Visual Translator tool to translate text phrases from procedures into one or more spoken languages.

Reporting Tools

Progress Report Builder Deployment Guide (Windows only)

An administration and development guide for generating Report Builder reports using the Progress Report Engine.

Progress Report Builder Tutorial (Windows only)

A tutorial that provides step-by-step instructions for creating eight sample Report Builder reports.

Progress Report Builder User's Guide (Windows only)

A guide for generating reports with the Progress Report Builder.

Progress Results Administration and Development Guide (Windows only)

A guide for system administrators that describes how to set up and maintain the Results product in a graphical environment. This guide also describes how to program, customize, and package Results with your own products. In addition, it describes how to convert character-based Results applications to graphical Results applications.

Progress Results User's Guide for Windows and *Progress Results User's Guide for UNIX*

Platform-specific guides for users with little or no programming experience that explain how to query, report, and update information with Results. Each guide also helps advanced users and application developers customize and integrate Results into their own applications.

4GL

Building Distributed Applications Using the Progress AppServer

A guide that provides comprehensive information about building and implementing distributed applications using the Progress AppServer. Topics include basic product information and terminology, design options and issues, setup and maintenance considerations, 4GL programming details, and remote debugging.

Progress External Program Interfaces

A guide to the external programming interfaces supported by Progress. This manual covers the Host Language Call (HLC) Interface, the system clipboard, named pipes, shared libraries and DLLS, Windows Dynamic Data Exchange (DDE), COM objects, ActiveX Automation, ActiveX controls, sockets, XML, SAX, and the SonicMQ 4GL Adapter.

Progress Internationalization Guide

A guide to developing Progress applications for markets worldwide. The guide covers both internationalization—writing an application so that it adapts readily to different locales (languages, cultures, or regions)—and localization—adapting an application to different locales.

Progress Language Reference

A three-volume reference set that contains extensive descriptions and examples for each statement, phrase, function, operator, widget, attribute, method, and event in the Progress language.

Progress on the Web

A manual that describes how to use the new WebClient, AppServer Internet Adapter, SmartObjects, and SonicMQ Adapter to create applications tailored for Internet, intranet, and extranet environments.

Progress Programming Handbook

A two-volume handbook that details advanced Progress programming techniques.

Database

Progress Database Design Guide

A guide that uses a sample database and the Progress Data Dictionary to illustrate the fundamental principles of relational database design. Topics include relationships, normalization, indexing, and database triggers.

Progress Database Administration Guide and Reference

This guide describes Progress database administration concepts and procedures. The procedures allow you to create and maintain your Progress databases and manage their performance.

DataServers

Progress DataServer Guides

These guides describe how to use the DataServers to access non-Progress databases. They provide instructions for building the DataServer modules, a discussion of programming considerations, and a tutorial.

Each DataServer has its own guide as follows:

- *Progress/400 Product Guide*
- *Progress DataServer for Microsoft SQL Server Guide*
- *Progress DataServer for ODBC Guide*
- *Progress DataServer for ORACLE Guide*

MERANT ODBC Branded Driver Reference

The Enterprise DataServer for ODBC includes MERANT ODBC drivers for all the supported data sources. For configuration information, see the MERANT documentation, which is available as a PDF file in *installation-path*\odbc. To read this file you must have the Adobe Acrobat Reader Version installed on your system. If you do not have the Adobe Acrobat Reader, you can download it from the Adobe Web site at: http://www.adobe.com/products/acrobat/readstep.html.

SQL-89/Open Access

Progress Embedded SQL-89 Guide and Reference

A guide to Progress Embedded SQL-89 for C, including step-by-step instructions on building ESQL-89 applications and reference information on all Embedded SQL-89 Preprocessor statements and supporting function calls. This guide also describes the relationship between ESQL-89 and the ANSI standards upon which it is based.

Progress Open Client Developer's Guide

A guide that describes how to write, build, and deploy Java and ActiveX applications, and Java applets that run as clients of the Progress AppServer. This guide includes information about how to expose the AppServer as a set of Java classes or as an ActiveX server, and how to choose an Open Client distribution package fro run time.

Progress SQL-89 Guide and Reference

A user guide and reference for programmers who use interactive Progress/SQL-89. It includes information on all supported SQL-89 statements, SQL-89 Data Manipulation Language components, SQL-89 Data Definition Language components, and supported Progress functions.

SQL-92

Progress Embedded SQL-92 Guide and Reference

A guide to Progress Embedded SQL-92 for C, including step-by-step instructions for building ESQL-92 applications and reference information about all Embedded SQL-92 Preprocessor statements and supporting function calls. This guide also describes the relationship between ESQL-92 and the ANSI standards upon which it is based.

Progress JDBC Driver Guide

A guide to the Java Database Connectivity (JDBC) interface and the Progress SQL-92 JDBC driver. It describes how to set up and use the driver and details the driver's support for the JDBC interface.

Progress ODBC Driver Guide

A guide to the ODBC interface and the Progress SQL-92 ODBC driver. It describes how to set up and use the driver and details the driver's support for the ODBC interface.

Progress SQL-92 Guide and Reference

A user guide and reference for programmers who use Progress SQL-92. It includes information on all supported SQL-92 statements, SQL-92 Data Manipulation Language components, SQL-92 Data Definition Language components, and Progress functions. The guide describes how to use the Progress SQL-92 Java classes and how to create and use Java stored procedures and triggers.

Deployment

Progress Client Deployment Guide

A guide that describes the client deployment process and application administration concepts and procedures.

Progress Developer's Toolkit

A guide to using the Developer's Toolkit. This guide describes the advantages and disadvantages of different strategies for deploying Progress applications and explains how you can use the Toolkit to deploy applications with your selected strategy.

Progress Portability Guide

A guide that explains how to use the Progress toolset to build applications that are portable across all supported operating systems, user interfaces, and databases, following the Progress programming model.

WebSpeed

Getting Started with WebSpeed

Provides an introduction to the WebSpeed Workshop tools for creating Web applications. It introduces you to all the components of the WebSpeed Workshop and takes you through the process of creating your own Intranet application.

WebSpeed Installation and Configuration Guide

Provides instructions for installing WebSpeed on Windows and UNIX systems. It also discusses designing WebSpeed environments, configuring WebSpeed Brokers, WebSpeed Agents, and the NameServer, and connecting to a variety of data sources.

WebSpeed Developer's Guide

Provides a complete overview of WebSpeed and the guidance necessary to develop and deploy WebSpeed applications on the Web.

WebSpeed Product Update Bulletin

A booklet that provides a brief description of each new feature of the release. The booklet also explains where to find more detailed information in the documentation set about each new feature.

Welcome to WebSpeed (Hard copy only)

A booklet that explains how WebSpeed software and media are packaged. *Welcome to WebSpeed!* also provides descriptions of the various services Progress Software Corporation offers.

Reference

Pocket Progress (Hard copy only)

A reference that lets you quickly look up information about the Progress language or programming environment.

Pocket WebSpeed (Hard copy only)

A reference that lets you quickly look up information about the SpeedScript language or the WebSpeed programming environment.

1

Overview Of the Progress Application Development Environment

The Progress Application Development Environment (ADE) provides access to a variety of tools that help you develop applications. This chapter provides an overview of the ADE Desktop and introduces you to the tools that comprise the ADE. In addition to a general description of the ADE and its tools, this chapter provides a more detailed description of the AppBuilder.

This chapter includes the following sections:

- Starting the ADE
- ADE Toolset
- Progress Online Help System
- Overview Of the AppBuilder

1.1 Starting the ADE

The ADE Desktop provides access to a number of ADE tools. To start the Desktop, choose Programs→ Progress→ Desktop from the Windows Start menu. An ADE Desktop window similar to the one shown in Figure 1–1 opens.



Figure 1–1: ADE Desktop

NOTE: The contents and appearance of the ADE Desktop depend upon the tools licensed and installed at your site. You might see fewer or more tools than those shown in Figure 1–1.

The ADE Desktop has a:

- **Menu bar** — Provides access to the available ADE Desktop menus. You use the Desktop menus to perform file operations, start ADE tools, minimize or maximize the Desktop, view messages, and access online help.

  For a description of the selections on the Help menu, see the "Progress Online Help System" section later in this chapter.

- **Toolbar** — Provides access to a set of icons that you use to easily access your ADE Tools. The individual tools are discussed in the "ADE Toolset" section.

NOTE: If you leave your cursor over an icon, a *ToolTip* appears naming the tool.

1.1.1 Starting a Tool From the ADE Desktop

You can start a tool from the ADE Desktop using either of the following methods:

- Clicking once on its icon
- Choosing a tool from the Tools menu

1.1.2 Starting a Tool From Another Tool

Most of the individual tools have their own Tools menu. When a tool has its own Tools menu, you can run other tools from that tool without returning to the ADE Desktop. For example, the AppBuilder has a Tools menu that includes the Data Dictionary option. As a result, you can run the Data Dictionary from the AppBuilder.

The option of accessing tools from other tools in this fashion helps you manage the number of tools running (you cannot have more than one instance of a tool running within a session), and also helps you establish a relationship between the tools you run. For example, if you run the Data Dictionary from the AppBuilder, you must exit the Data Dictionary before returning to the AppBuilder.

An exception to this behavior is PRO*Tools (described in the "ADE Toolset" section). When you run PRO*Tools, it displays as a palette of icons that are available for use at any time.

1.2 ADE Toolset

The descriptions of the ADE tools in this section are listed in the order in which they appear in the Tools menu. An asterisk (*) indicates that you can also access the tool from a tool icon. If a tool has not been licensed or installed at your site, its icon does not appear in the ADE Desktop.

The ADE tools are:

- **Data Dictionary*** — A tool used to define the structure of your database including tables, fields, and indexes, and to generate reports about the database. You can use this tool to define application defaults that, for example, validate data, perform security checks, and generate unique sequence numbers. The Progress 4GL gives you the flexibility to override most of these defaults in individual applications.

  You also use the Data Dictionary to create and connect to databases.

- **Procedure Editor*** — An editor used to create, edit, compile, and run unstructured 4GL procedures (as opposed to the structured procedures created by the AppBuilder). The Procedure Editor also provides functionality for checking syntax, managing multiple files, and customizing its environment.

- **Data Administration** — A tool used to perform a variety of database administration tasks including creating, starting, and shutting down databases, dumping and loading database files, importing and exporting data, and implementing database security.

- **PRO*Tools** — A set of utility programs to aid in developing and running applications in the Progress ADE. For example, one of the PRO*Tools allows you to edit your PROPATH, the path Progress searches to find programs. When you select PRO*Tools from the Tools menu, the ADE displays a resizable palette of icons. You then run an individual PRO*Tool by clicking on its icon.

- **AppBuilder*** — The primary visual programming tool in the Progress ADE. All of the work of fabricating and assembling application components is done in the AppBuilder. The AppBuilder's Section Editor adds convenience to editing 4GL code while preserving the standardized structure that works best with Progress application components, such as basic objects and SmartObjects.

- **Web Tools** — Tools used to monitor your Web server environment, manipulate objects such as deleting and checking syntax, viewing database schema information, and issuing operating system commands through an intuitive browser interface that also provides links to sample applications and support services.

- **Report Builder*** — A robust data access and reporting tool that allows you to create production-quality reports that select, analyze, and present data from your database tables in a variety of ways. Report Builder's presentation features give you the ability to control fonts and colors, and to include images in your reports. The report definitions are saved in a report library.

- **Results*** — An interactive menu-driven tool that lets end users with little or no programming experience query, report on, and maintain information stored in database tables. As an application developer, you can also customize Results according to the needs of individual sites, and then integrate it into your applications.

- **Translation Manager*** — A tool used to manage the process of customizing an application for specific end-user languages. Translation Manager allows you to select words and phrases from source procedure files for translation into other languages. These words and phrases are assembled into translation kits that you can send to translators who perform the actual translation. After you receive the completed translations, you use the Translation Manager to review the translations and integrate them into your applications. The Translation Manager also provides statistical information about your translation projects.

- **Visual Translator** — A tool used by the translators who translate the actual words and phrases into other languages. The Visual Translator uses a translation kit that was generated by the Translation Manager, and allows the translator to work either from a list or in the visual context of an application's windows.

- **Application Compiler** — A tool used to compile individual source procedures or a group of procedures to produce executable code that is permanently stored.
- **Application Debugger*** — An interactive tool used to monitor the execution of an application by stopping and restarting it at specified lines. You can also display or update information about the procedure you are running (such as the names and values of variables) without modifying any procedure code. The Debugger runs as a separate process with its own window.
- **Actuate e.Report Designer Professional *** — A 32-bit Windows graphical tool used to design, build, and test Actuate components and report designs within an object-oriented development environment. The Actuate e.Report Designer Professional contains ActiveX controls and APIs (formerly the Actuate SDK) that allow Actuate reporting functionality to be embedded into Progress applications.

1.3 Progress Online Help System

Progress provides an online help system that provides information about using the ADE tools, reference material on the Progress 4GL, Cue Cards for the SmartObjects, and explanatory information on Progress error messages. The Progress online help system contains information for both programming and system administration audiences.

The Progress online help system uses hypertext links to access help topics. In addition to providing help information, help topics provide links to other help systems and information. Hypertext links appear on your screen as colored text with a solid underline. To go to the topic associated with the link, click the link text. After viewing the help, you can return to the previous screen by choosing the Back button.

You can access help information from the Progress Desktop in the following ways:

- Press **F1** to get *context-sensitive help* about the current window or dialog box. Most dialog boxes in the ADE toolset also have a Help button that provides access to help information.
- Use the Help menu to access information about the current tool or the online reference guide. The Help system contains topics that describe what information you can access through each of the options on the Help menu.

1.4 Overview Of the AppBuilder

This section provides an overview of the AppBuilder tool and includes information on starting the AppBuilder and a description of the AppBuilder windows, menu options, and the AppBuilder Object Palette. The AppBuilder is a visual programming environment that allows you to rapidly build complex applications. After you read this section, you should be familiar with the components of the AppBuilder and how to work with the AppBuilder and Object Palette options. For more detailed information about using the AppBuilder and Object Palette, see the *Progress AppBuilder Developer's Guide*.

1.4.1 Starting the AppBuilder

To start the AppBuilder, choose the AppBuilder tool icon in the ADE Desktop. The AppBuilder main window opens with its Object Palette, as shown in Figure 1–2.

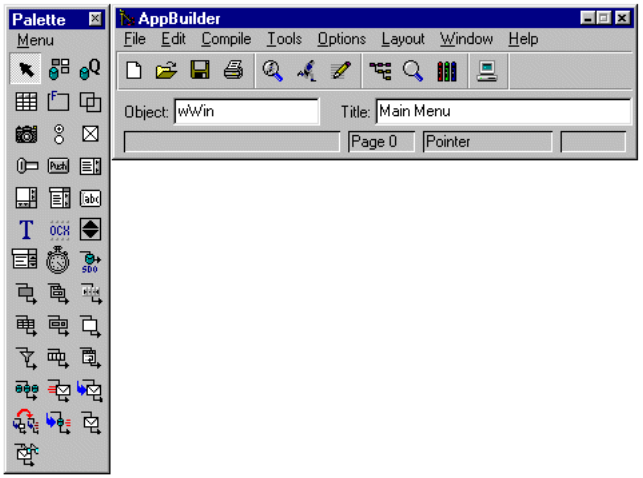


Figure 1–2: AppBuilder Main Window and Object Palette—Initial Display

NOTE: The components in the Object Palette might differ from those shown in Figure 1–2, depending upon your license. Also, if your screen resolution is not standard SVGA (800 x 600 pixels), the appearance of the Object Palette might be different.

1.4.2 AppBuilder Main Window

When you start the AppBuilder, the main window and the Object Palette open. You can move these two windows independently on your screen. The Object Palette has several display options that do not depend on the AppBuilder main window. For a discussion of these options, see the "AppBuilder Object Palette" section of this chapter.

The AppBuilder main window has four elements, as shown in Figure 1–3.

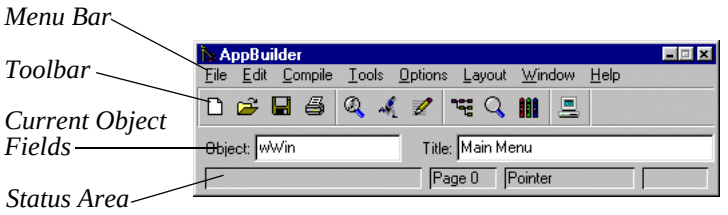


Figure 1–3: The AppBuilder Main Window

The following sections describe the four elements of the AppBuilder main window.

Menu Bar

The menu bar provides access to most of the AppBuilder functions. Some menu options are not available when working in WebSpeed development mode:

- **File** — Provides access to file and window operations such as New, Open, Close, Save, and Print. You can also use this option to view and select from a list of most recently used files. The Most Recently Used File List option is enabled by default, and you can specify the number of files you want to appear in this list through the Options→ Preferences menu item.
- **Edit** — Provides access to general editing operations such as copying, deleting, and pasting objects. In addition to these general operations, the Edit menu provides options for copying and inserting files, specifying the tab order for a frame, and accessing a specific page.
- **Compile** — Provides access to a number of options you can use to run, view, test, and debug your code.
- **Tools** — Provides access to a number of AppBuilder tools, such as the Data Dictionary and Procedure Editor, and a number of other options to help you design and build applications, such as a Color Editor and Property Sheets.
- **Options** — Provides access to AppBuilder preference options including colors and grid alignments.

- **Layout** — Provides access to options that you use to align objects using predefined layouts or customized layouts.
- **Window** — Provides access to the Section Editor, properties windows, and editors.
- **Help** — Provides access to help information, including error message explanations and Cue Cards for SmartObjects.

As you complete the tutorial exercises in this guide, you will become familiar with many of the menu selections. The menus and icons of the AppBuilder main window also provide access to all of the functions available in the ADE Desktop, including the ability to start other tools and to access the Progress online help system.

For more detailed information about the AppBuilder menu bar options, see the online help for that menu or the chapter on the AppBuilder interface in the *Progress AppBuilder Developer's Guide*.

Toolbar

Each icon in the AppBuilder toolbar provides quick access to frequently used menu options. Table 1–1 describes the function of each icon.

Table 1–1: Toolbar Icons *(1 of 3)*

Icon	ToolTip	Description
	New	Creates a new object file and displays an associated design window or tree view. The New icon on the toolbar corresponds to the File→ New menu option. The AppBuilder creates a *design window* or displays a tree view when you create a new object or open an existing procedure file. The design window is analogous to a canvas on which you visually assemble all or part of an application. For each action you take that modifies the appearance or functionality of the object, the AppBuilder adds, deletes, or modifies the 4GL code contained in the object's procedure file
	Open	Opens an existing procedure file and displays its associated design window. The Open icon corresponds to the File→ Open menu option.
	Save	Saves the current procedure file. The Save icon corresponds to the File→ Save menu option.

Table 1–1: Toolbar Icons *(2 of 3)*

Icon	ToolTip	Description
	Print	Prints the currently selected procedure. When you select the Print icon, a Windows Print dialog box opens. From this Print dialog box, you can specify printing options and properties. By default, the Print dialog box is enabled. You can use Options→ Preferences on the AppBuilder menu bar to disable the Print dialog box or to specify preferences for the page length and font. For more information on enabling the Windows Print dialog box and setting preferences, see the *Progress AppBuilder Developer's Guide*.
	Procedure Settings	Displays and allows editing of the procedure settings for the current procedure file. The Procedure Settings icon corresponds to the Tools→ Procedure Settings menu option. *Procedure settings* maintain information about procedure files, the attributes they possess, and standardized ways in which they interact with other procedures.
	Run	Compiles and runs the current procedure file. The Run icon corresponds to the Compile→ Run menu option.
	Stop	Stops the currently running procedure. The Stop icon replaces the Run icon when you run a procedure.
	Edit Code	Opens the Section Editor, which enables you to edit certain sections of the procedure file. The Edit Code icon corresponds to the Window→ Code Section Editor menu option.
	List Objects	Opens the List Objects dialog box. This dialog box lists all of the objects currently in use in any design window. You can access an individual object's property sheet by double-clicking an object in the list.
	Object Properties	Opens the property sheet for the current object. A property sheet gives you control over settings that pertain to individual objects, including fonts, geometry (window size), and other window settings. The Object Properties icon corresponds to the Tools→ Property Sheet menu option.

Table 1–1: Toolbar Icons *(3 of 3)*

Icon	ToolTip	Description
	Colors	Opens the Choose Color dialog box. Use this dialog box to specify a foreground and background color for the selected objects. The Colors icon corresponds to the Tools→ Color menu option.
	Switch to Remote/Local Development	Changes the development mode between local (client/server) mode and remote (WebSpeed) mode. The tutorial exercises require that you run in local mode. For more information on using the remote mode, see the *WebSpeed Developer's Guide*. **NOTE:** You might not see this icon, depending upon your license.

Current Object Fields

The current object fields display information about the *current object*, which is a single object in a design window that has design focus. If you select multiple objects in a design window, then no current object information is displayed.

The current object fields display the following information:

- **Object** — Specifies the name of the current object.
- **Title/Label/Text/Master/File/OCX** — For a basic object, this field specifies the object's title or label, if it has one. Text objects, for example, have no label or title, so for a text object this field displays its contents.

  For SmartObject instances, which are based on prefabricated masters stored in separate files, this field displays the object's title bar text (SmartWindows and SmartDialogs) or the pathname of the master file (all other SmartObjects).

  For an ActiveX (OCX), this field displays the name of the ActiveX object.

Status Area

From left to right, the four fields in the status area display are:

- **File status** — Information about current AppBuilder processing.
- **Page status** — The current page (applicable to SmartContainers only).
- **Object Palette status** — The type of object currently selected on the Object Palette.
- **Lock status** — For inserting objects. For more information on lock status, see the "How the Object Palette Responds To the Mouse" section in this chapter.

1.4.3 AppBuilder Object Palette

From the Object Palette, you can select basic objects, SmartObjects, and ActiveX (OCX) controls to insert into a design window. You can also create and insert new SmartObjects from the Object Palette.

Object Palette Options

The Object Palette's menu options allow you to control some characteristics of the Object Palette itself. For example, when you choose Menu→ Options→ Show Menu Only, the icons are hidden and only the Menu options appear in the Object Palette. This option is useful when working with a VGA resolution and you do not want to use valuable screen space to display the palette's icons.

The other two Object Palette options give you control over other aspects of its appearance. If you choose Menu→ Options→ Top-Only Window, the Object Palette stays on top of all other windows, including design windows. If you choose Menu→ Options→ Save Palette, the AppBuilder saves the options, position, and size of the Object Palette for future sessions in the registry or in the `progress.ini` file.

You can resize the Object Palette and move it to a more convenient area on your desktop or minimize it until you are ready to begin assembling your applications. To view an icon's name, place the cursor over the icon to display the ToolTip.

The icons in the bottom rows of the Object Palette represent SmartObjects, as shown in Figure 1–4. With the exception of the Pointer, which represents no object, all other icons on the Object Palette represent basic objects, which are also referred to as widgets.



Figure 1–4: The AppBuilder Object Palette

Table 1–2 lists the icons on the Object Palette and their functions. For more information about the objects in the Object Palette, see the *Progress AppBuilder Developer's Guide*, the *Progress ADM 2 Guide*, and the *Progress ADM 2 Reference*.

Table 1–2: Object Palette Icons *(1 of 4)*

Icon	ToolTip	Description
	Pointer	Selects an object in the design window or frame.
	DB-Fields	Adds a database field and a query for that field.
	Query	Adds a query and starts the QueryBuilder. The QueryBuilder widget is visible only at design time.
	Browse	Adds a browser.
	Frame	Adds a frame. Frames are used to visually and functionally group other objects.

Table 1–2: Object Palette Icons *(2 of 4)*

Icon	ToolTip	Description
	Rectangle	Adds a rectangle. Rectangles are used to visually group other objects.
	Image	Adds a static image.
	Radio-Set	Adds a radio button set. This object creates a three-button radio set with generic text.
	Toggle-Box	Adds a single check box with a generic label.
	Slider	Adds a slider. You can define sliders as horizontal or vertical.
	Button	Adds a pushbutton with a generic label.
	Selection-List	Adds a scrollable selection list.
	Editor	Adds a multi-line text editor.
	Combo-Box	Adds a non-OCX combo box.
	Fill-In	Adds a single-line editor.
	Text	Adds text to a window. You can specify a font and color for the text.
	OCX	Adds an ActiveX control.
	CSSpin	Adds a Progress Spin Button control.

Table 1–2: Object Palette Icons *(3 of 4)*

Icon	ToolTip	Description
	CSComboBox	Adds a Progress OCX combo box.
	PSTimer	Adds a Progress OCX Timer object.
	SmartDataObject	Adds a SmartDataObject (SDO) I/O interface. SmartDataObjects manage the data stream to and from disk, filter records, and respond to requests from other SmartObjects.
	SmartObject	Adds a generic ADM Smart template. You use this template to create new types of SmartObjects.
	SmartFolder	Adds an ADM tabs widget. Tabs widgets are used in SmartFolders to provide a context in which you can visually and functionally divide an interface.
	SmartPanel	Adds one of the predefined ADM button arrays. SmartPanels provide upstream control of other SmartObjects.
	SmartDataBrowser	Adds an ADM data browser. SmartDataBrowsers provide a multi-record row/column interface to data.
	SmartDataViewer	Adds an ADM data viewer. SmartDataViewers provide a single-record interface to data that you can arrange for a specific design.
	SmartDataField	Adds customized capabilities to a SmartDataViewer.
	SmartFilter	Adds an ADM Smart Query-By-Form.
	SmartToolBar	Adds an ADM menu and toolbar.

Table 1–2: Object Palette Icons *(4 of 4)*

Icon	ToolTip	Description
	SmartContainer	Adds an outer-level container object, such as a SmartWindow, to another container, imposing a parent/child relationship on objects that do not normally inherit from each other.
	SmartBuisnessObject	Adds a special ADM SmartContainer that organizes the data streams of several SmartDataObjects.
	SmartProducer	Adds an ADM message handler that creates messages to be transmitted by a Progress SonicMQ message service.
	SmartConsumer	Adds an ADM message handler that receives messages transmitted by a Progress SonicMQ message service.
	SmartRouter	Adds an ADM message handler that routes incoming messages from a SmartConsumer to appropriate SmartB2BObjects.
	SmartB2BObject	Adds an ADM transformation object that transforms information to and from XML, based on a protocol shared between the message's sender and recipient.
	SmartSender	Adds a customizable ADM object that transforms information acording to your rules, passing the result to a SmartProducer.
	SmartReciever	Adds a customizable ADM object that recieves information from a SmartConsumer and then transforms it acording to your rules.

In addition to choosing an object from the Object Palette, you can choose an object from the Object Palette menu. When you choose an object from the menu, some options include specific properties for the object. For example, if you choose an Editor object from the menu, you have the additional option of choosing a Read-Only Editor object.

You can add OCX icons or submenu options to the Object Palette by choosing Menu→ Add OCX and then choosing Add as Palette Icon or Add to Palette SubMenu. Specify the appropriate OCX file information. The icon or submenu option appears on the Object Palette.

Customizing the Object Palette

In addition to providing a variety of display options for the Object Palette, the AppBuilder allows you to customize its contents. Choose Menu→ Use Custom to add to or modify the files that determine the Object Palette's behavior and appearance.

For example, if you create new SmartObject templates, you can add them to the Object Palette. You can also change certain default settings for the standard SmartObject templates.

NOTE: Creating a new SmartObject template is different from fabricating a SmartObject master from an existing template, as you do in Chapter 4, "An Overview Of SmartObject Technology."

How the Object Palette Responds To the Mouse

The icons on the Object Palette respond differently to the left and right mouse buttons:

- **Left mouse button** — When you click the left button, the AppBuilder enters insertion mode immediately for basic objects and the SmartFolder. For other SmartObjects, the AppBuilder displays a dialog box from which you can choose a SmartObject master. After you select a SmartObject the AppBuilder enters insertion mode. When you choose the **OCX** icon, the AppBuilder displays a dialog box from which you choose an ActiveX control.

  You can lock an Object Palette icon to insert multiple instances of an object. To lock an icon for SmartObjects or ActiveX controls, click it once. If there is more than one object of this type available, select one from the Choose dialog box that opens. After the dialog box closes, a click the object's icon again before inserting to lock it. To lock an icon for other objects on the palette, click twice on an object. A lock image appears over an icon when you lock it. Choose the Pointer object to remove the lock.

- **Right mouse button** — When you click the right button, the AppBuilder displays a pop-up menu for the object you chose. The options available on the pop-up menu depend on the selected object.

  For basic objects, the menu allows you to choose from among the available variations of the object; if there is only one type of that object available, the menu has only the Default option. This also applies to the SmartFolder.

  For SmartObjects, with the exception of the SmartFolder, the pop-up menu generally allows you to choose from among any available prefabricated SmartObject masters, or to create a new master from one of the SmartObject templates provided with the product. Instead of allowing you to create a new master, the pop-up menu for the SmartPanel allows you to create a new instance of one of the four SmartPanel templates.

The pop-up menu for the SmartContainer has a Choose SmartContainer option. You use this option to select an existing container, such as a SmartWindow, SmartFrame, or SmartDialog. To create a new SmartContainer, choose the New icon on the AppBuilder toolbar and select the container you want from the New dialog box.

1.4.4 Advanced Editing Features For Progress Editors

If your Progress installation runs under a development license, you have access to the following advanced editing features:

- **Color coding** — Also called color syntax-highlighting, color coding is a powerful feature in which a Progress editor displays various types of 4GL syntax elements in a Progress source file such as keywords, comments, and character strings in distinct colors.

  For example, all keywords might appear in blue. Color coding makes it easy to understand the syntax and structure of a 4GL program. You can customize the color coding used for language elements and/or change their style, for example, to apply a bold or italic typeface.

  Progress automatically supports color coding for the Progress 4GL and for many other programming and scripting languages, including HTML, C/C++, Perl, and Java.

- **Embedded color coding** — Embedded color coding recognizes multiple languages in the same file and can display them with their own color-coding schemes. Using embedded color coding simplifies editing source files that contain code in multiple languages: the editor displays each language's code using a distinct color-coding scheme and all other language-specific features.

  By default, the editors recognize only embedded SpeedScript, JavaScript, VBScript, and/or Java in HTML files.

- **Syntax expansion** — Syntax expansion in Progress editors occurs when you type a 4GL keyword, statement, or block structure and then press the **SPACEBAR**. The editor expands the syntax of the keyword or other language element by inserting a template specific to it.

- **Alias expansion** — Alias expansion allows you to define shortcuts that extend or modify syntax expansion in Progress editors. You can use the aliases to insert more complete code and syntax templates when you type the shortcut and press the **SPACEBAR**.

  For example, when you type in **DISP**, then press the **SPACEBAR**, the editor by default does not expand this syntax. However, you can use alias expansion to define a code template to insert more of the statement's syntax. Then when you type **DISPLAY** and press the **SPACEBAR**, the editor expands to the template you defined.

  Progress provides a number of default aliases. These aliases are in the file `p4gl.als` located in your working editor configuration directory. By default, this is the PROEDIT directory in your Progress working directory. Whenever you use the Alias Editor dialog box to modify or add aliases for Progress source files, the editor saves the changes to the `p4gl.als` file.

- **Syntax indenting** — Syntax indenting in Progress editors occurs when you press the **ENTER** key at the end of a block statement. The editor inserts a new line and indents the cursor relative to the block statement.

  For example, if you create a Progress source file with the cursor positioned at the end of a line that contains the DO keyword, and you press the **ENTER** key, the editor inserts a new line and indents the cursor relative to the DO keyword.

- **SmartPaste** — The SmartPaste feature allows you to paste lines of text into a Progress source file and indent them relative to the surrounding code. This feature saves coding time by making it unnecessary to indent pasted code manually. It is especially useful when you copy and paste code among several Progress source files.

  For example, if you select several lines of code in a Progress source file, copy them to the clipboard using **CTRL-C**, then paste them inside a FOR block using **CTRL-V**, the Progress editors use SmartPaste to indent the pasted lines correctly relative to the indentation level of the FOR loop.

For more information about the advanced editing features, see the *Progress AppBuilder Developer's Guide* and the Procedure Window and Common Dialog help system.

Now that you are familiar with the ADE Desktop and the AppBuilder main window, you are ready to learn more about developing applications using the Progress tools and SmartObject technology. To help you learn, this guide introduces the SportsPro sample application, which was designed to highlight these features.

2

Working With the Sample Application

This chapter provides information on running and using the SportsPro sample application. This application is included when you perform a complete installation of Progress. The sample application highlights useful programming techniques and solutions using SmartObjects and the Application Development Model (ADM2) for a distribution business.

In this chapter, you will focus on learning the application's Order Tracking System from an end user's perspective. In later chapters, you will build its user interface and some of its SmartObject components.

For more information about SmartObject technology, see Chapter 4, "An Overview Of SmartObject Technology."

This chapter also briefly examines the other functions of the SportsPro sample application. When you are ready to look deeper into the sample application, follow the pointers provided in this chapter to learn why the application is coded the way it is.

This chapter includes the following sections:

- The SportsPro Sample Application
- Creating and Connecting To the Sports2000 Database
- Copying the Sample Files
- Compiling the Sample Files
- Running the Sample Application
- Working With the Order Tracking System

2.1 The SportsPro Sample Application

The SportsPro sample application was created to show how you can quickly build complex applications with SmartObjects. As it evolves, it demonstrates more techniques for effectively using Progress products to solve a variety of business needs. The SportsPro sample application has something to interest you, no matter what environment you plan to develop applications for. You can run it as a stand-alone application, distribute it with an AppServer, or use the WebClient to run it across an Internet, intranet, or extranet connection.

As you become more familiar with the Progress products, the sample application is a good place to explore for ideas on how to add functionality to your own applications. After you run the sample application, open its modules in the AppBuilder. Examining the objects in the AppBuilder and the Section Editor provides you with insights on using SmartObjects to build your own applications.

2.2 Creating and Connecting To the Sports2000 Database

Before running or assembling the sample application described in this chapter, you must create a copy of the Sports2000 database and connect to it.

To complete this task, you use the Data Dictionary tool. The Data Dictionary tool is part of the Application Development Environment (ADE) and is used to create and connect to databases and to edit the definitions of tables, fields, indexes, and other database-related information.

NOTE: When you create a copy of the Sports2000 database, you specify a database name and a location where you want to create the copy. When prompted for the location, use the working directory you defined when you installed the product. By default, this directory is `C:\Progress\wrk\`.

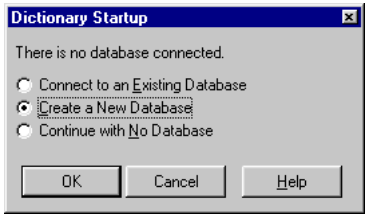
Making a copy of the Sports2000 database allows you to make changes to the copy but preserves the original database that you installed with your program files. In addition, you can delete your current copy and make a new copy, or you can create additional copies of the Sports2000 database in the same directory using the Replace If Exists option.

Follow these steps to create and connect to your copy of the Sports2000 database:

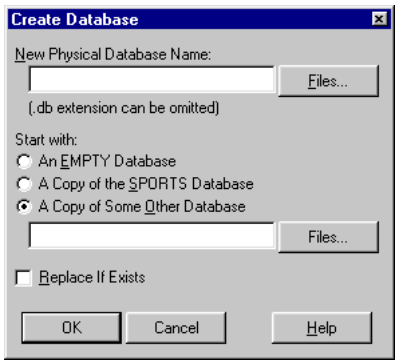
1 ♦ From the Windows Start menu, choose Programs→ Progress→ Desktop.

2 ♦ From the ADE Desktop, choose the Data Dictionary tool icon. 

3 ♦ When you run the Data Dictionary, it looks for a connected database. If there is no connected database, the Dictionary Startup dialog box opens:



4 ♦ The default selection for the Dictionary Startup dialog box is Connect to an Existing Database. For this exercise, select Create a New Database and choose OK. The Create Database dialog box opens:



5 ♦ Type the path and new name for the new physical database. For example: **c:\progress\wrk\sports2000.db**.

6 ♦ The default selection for the Create Database dialog box is An EMPTY Database. For this exercise, select the radio button to create a copy of some other database.

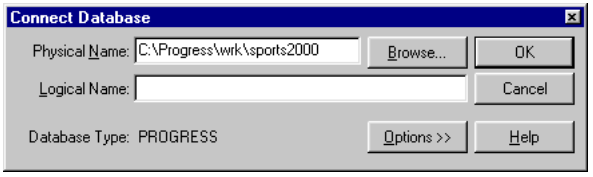
NOTE: You should place your copy of the new database in your working directory. By default, this is `c:\progress\wrk`.

7 ♦ Choose the Files button, browse to the location of the original Sports2000 database, and select it.

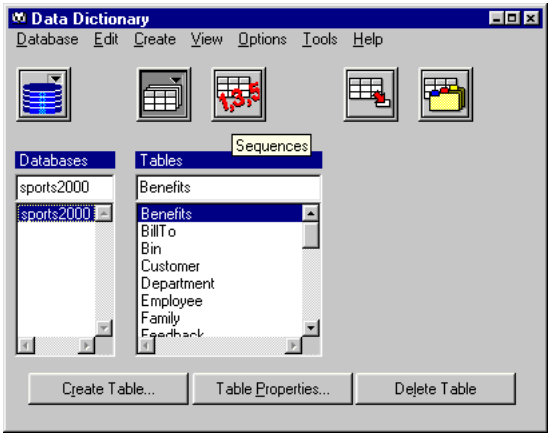
NOTE: The Sports2000 database is located in your Progress installation directory. By default, this is `c:\program files\progress`.

8 ♦ Choose OK.

The Data Dictionary creates a copy of the Sports2000 database in your working directory. The Connect Database dialog box opens and displays the name of the database you just created:



9 ♦ Choose OK in the Connect Database dialog box. The Data Dictionary makes the connection and its startup window opens:



You have now created and connected to your working copy of the Sports2000 database. The Data Dictionary window lists the Sports2000 database as connected and the tables in the database.

The icons in the toolbar govern what you can do with the Data Dictionary and what information displays for the selected database. When you choose an icon, buttons for varying tasks appear at the bottom of the window. Table 2–1 lists the icons and their associated displays and tasks.

Table 2–1: Data Dictionary Icons

Icon	Displays	Available Task Buttons
	Connected databases	Create Database Database Properties
	Tables in selected database	Create Table Table Properties Delete Table
	Sequences in selected database	Create Sequence Sequence Properties Delete Sequence
	Fields in selected table	Create Field Field Properties Delete Field
	Indexes on selected table	Create Index Index Properties Delete Index

10 ♦ Use the Data Dictionary to familiarize yourself with the structure of the Sports2000 database.

11 ♦ When you are ready, choose Database→ Exit to return to the ADE Desktop.

2.3 Copying the Sample Files

After creating a copy of the Sports2000 database and connecting to it, copy the tutorial sample files to your working directory and compile them. This enables you to experiment with the tutorial sample files while preserving the original files.

By default, the sample files are installed to `program files\progress\src\sports2000\gui` and `program files\progress\src\sports2000\images`. Copy the files in the `sports2000\gui` directory to your working directory. Then create an `images` subdirectory under your working directory and copy the files in `sports2000\images` to that subdirectory.

2.4 Compiling the Sample Files

To make your sample files run more efficiently, use the Application Compiler to compile the sample files you copied. The Application Compiler names the compiled files using a `.r` file extension.

Follow these steps to compile the sample files:

1 ♦ Choose Tools→ Application Compiler from the menu. The Application Compiler dialog box opens.

2 ♦ Choose Start Compile. The Compiler Results dialog box opens showing the AppBuilder's progress compiling the sample files.

NOTE: You do not need to specify a path because the Application Compiler looks for the files in your working directory.

3 ♦ Choose OK to return to the Application Compiler.

4 ♦ Choose File→ Exit to close the Application Compiler.

You have created a copy of the Sports2000 database, connected to your database, copied the sample application files, and compiled them. You can now run the sample application.

2.5 Running the Sample Application

Running the SportsPro sample application gives you a feel for building applications with the Progress AppBuilder and SmartObject technology. The sample application contains several modules, each demonstrating a technique for solving a business problem. After viewing the sample application, you will build part of the Order Tracking System module in the tutorial and exercises later in this guide.

Follow these steps to run the SportsPro sample application:

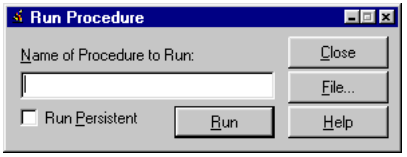
1 ♦ If you have not already connected to the database, connect to your copy of the Sports2000 database. For information about connecting to your copy of the Sports2000 database, see the "Creating and Connecting To the Sports2000 Database" section in this chapter.

2 ♦ Start the ADE Desktop from the Windows Start menu and then start the AppBuilder from Desktop toolbar.

3 ♦ Run PRO*Tools by choosing Tools→ PRO*Tools from the AppBuilder main window. The PRO*Tools palette opens:



4 ♦ Choose the Run button from the PRO*Tools palette. The Run Procedure dialog box opens:



5 ♦ Type **wmain.w** in the Name of Procedure to Run text box.

The `wmain.w` file is a window file containing the 4GL commands that create the window. If you copied your sample files as described in the "Copying the Sample Files" section, you do not need to qualify the filename with its directory. The `wmain.w` file should be in your working directory, and your working directory is a standard entry in your PROPATH.

6 ♦ Choose Run. The main window of the SportsPro sample application opens:



The SportsPro application's main window is a SmartWindow. The buttons on this SmartWindow use the new FLAT–BUTTON attribute to create a two-dimensional look.

Table 2–2 lists the components used in the main window of the SportsPro sample application. It lists all the procedure files, all the SmartObjects, and any simple objects with important functions. Except for SmartObjects like SmartFilters for which the Application Development Model (ADM2) supplies the masters, the procedure files and SmartObject master files listed in this table are in the `gui` folder in your working directory.

Table 2–2: Components Of the SportsPro Main Window *(1 of 2)*

Object Or File Name	Object Type	Function
`wmain.w`	SmartWindow	Serves as a container for SDOs referenced by two of the modules and the buttons that launch the modules.
`dcust`	SmartDataObject	Queries the Customer table to retrieve all customer records for the selected query. The query's handle is passed into one of the modules.
BUTTON-1	Button	Launches `wfndcust.w`, passing it the handle for `dcust`.

Table 2–2: Components Of the SportsPro Main Window *(2 of 2)*

Object Or File Name	Object Type	Function
BUTTON-2	Button	Launches `wquery.w`.
BUTTON-3	Button	Launches `wdragdrop.w`.
`dorder`	SmartDataObject	Queries the Order table to retrieve all order records for the selected query. The query's handle is passed into one of the modules.
BUTTON-4	Button	Launches `wfndord.w`, passing it the handle for `dorder`.
BUTTON-5	Button	Launches `wmntitem.w`.
BUTTON-6	Button	Launches `wordupd.w`.
BUTTON-7	Button	Launches `suppub.p`.
BUTTON-8	Button	Launches `gb2bintro.w`. If you choose OK in the dialog box, it runs `wstore.w`.
BUTTON-9	Button	Launches `worder.w`.

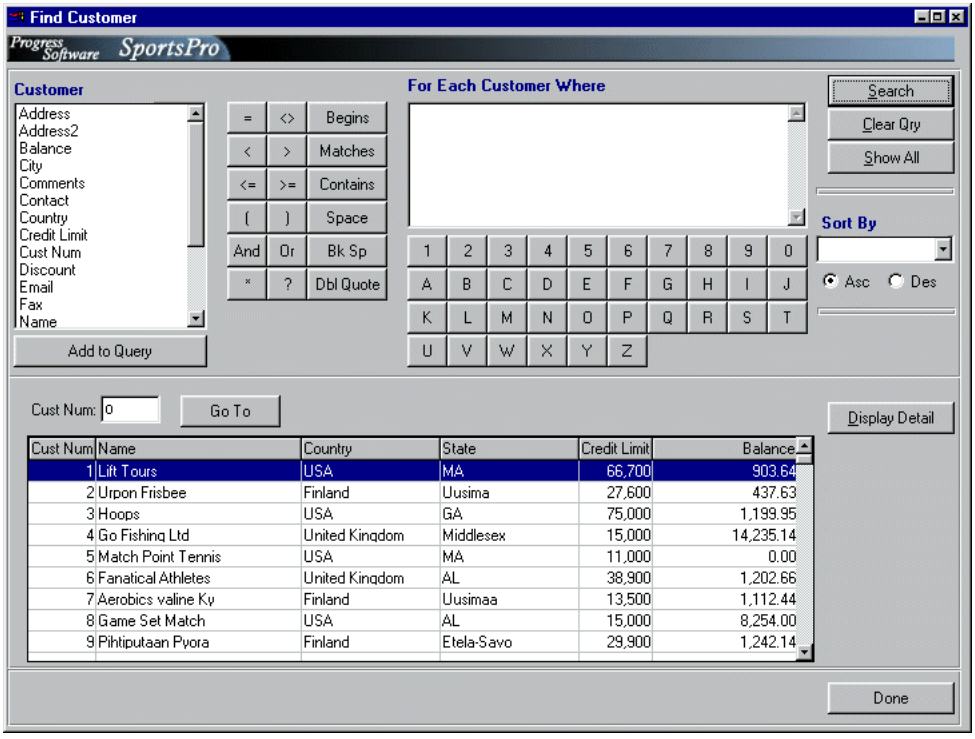
NOTE: Table 2–2 and the tables in later sections present the objects and procedures from each module in an event-driven order. The objects contained in another object, and the procedures executed in that object, are listed directly after that object. Where data passes between objects and procedures, they are listed chronologically. Otherwise, objects are listed from top to bottom in their container.

Each of the buttons on the sample application's main window launches a module. You can now examine these modules in turn. As you become more familiar with the Progress products, you can return to these modules and examine them in greater detail. These modules show how to accomplish several common programming tasks.

2.5.1 Dynamic Query Module

Before Progress Version 9, you had to define the selection criteria at compile time. You could achieve limited flexibility by using IF...THEN...ELSE or CASE statements. Progress Version 9 introduced new language elements that enable you to build dynamic queries. With the dynamic query structure, you do not have to anticipate every case for which an end user might want a query. At run time, end users can design their own queries to fit exactly the questions they need to answer.

Choose the Dynamic Query button on the SportsPro main window to launch this module:



When you launch the Dynamic Query module, it accepts the handle of a SmartDataObject on the SportsPro main window to serve as the data source for the query. As you create a query using the buttons in the upper section of this window, the module records the contents of the For Each Customer Where editor as a variable. Choosing the Search button launches the module's Search procedure. The Search procedure accepts the handle of the user input variable and uses it to replace the SmartDataObject's WHERE and SORT clauses.

Table 2–3 describes the major objects used in the Dynamic Query module.

Table 2–3: Components Of the Dynamic Query Module *(1 of 2)*

Object Or File Name	Object Type	Function
`wfndcust.w`	SmartWindow	Serves as a container for the other objects in the module.
`dcust`	SmartDataObject	Queries the Customer table to retrieve all customer records for the selected query. When this module is launched, it accepts the handle for the instance of `dcust` in the SportsPro main window. **NOTE:** Because this module uses a handle passed in from the main window, you must run the main window and launch it from there.
`getnewfields.p`	Procedure file	Runs during the initializeObject internal procedure to populate the lcustfieldname selection list.
lcustfieldname	Selection-list	Lists the fields in the Customer table that can be added to the query.
qrystringdisp	Editor	Displays the query being built. Its contents are passed to the Search internal procedure.
btnsearch	Button	Runs the Search internal procedure to display the records matching the query shown in the qrystringdisp editor.
sortfld	Combo-Box	Lists the fields by which the result set can be sorted.
lsort	Radio-set	Sets whether the result set is sorted in ascending or descending order.
bcust	SmartDataBrowser	Enables the user to select a customer record based on the query shown in the editor and the sort criteria.
btndspcust	Button	Launches `wdspcust.w`.

Table 2–3: Components Of the Dynamic Query Module *(2 of 2)*

Object Or File Name	Object Type	Function
wdspcust.w	SmartWindow	Serves as a container for the other objects in the Display Detail section of the module.
vcust2	SmartDataViewer	Displays full details of the customer currently selected in bcust.
stxtfld	SmartDataField	Displays the Comments field in vcust2 in a Cut/Copy/Paste editor.

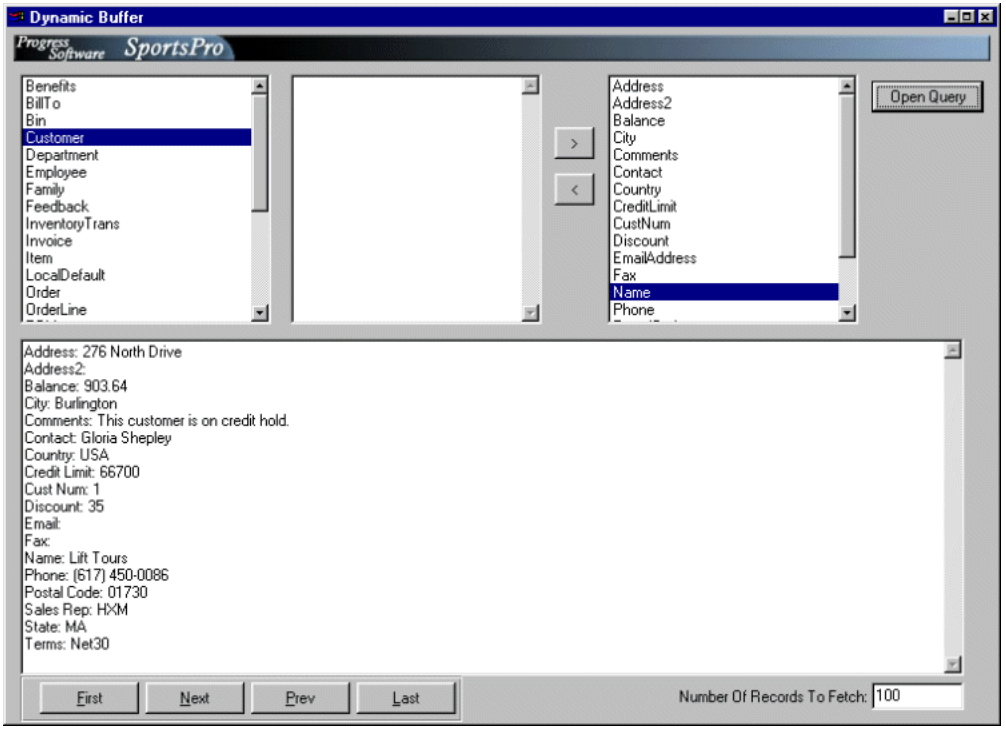
NOTE: The Dynamic Query module also runs when you choose the Find Customer button in the Customer section of the SportsPro Order Tracking System.

For more information on building dynamic queries, see the chapter on using dynamic widgets in the *Progress Programming Handbook*.

2.5.2 Dynamic Buffer/Temp Table Module

Progress puts the records that it reads into a buffer. Once a record is in the buffer, its fields are available to your application. Before Progress Version 9, you set the table for that buffer at compile time and could not change it at run time. Progress Version 9 introduced new language elements that enable you to build dynamic buffers. With a dynamic buffer, you can select the table for your buffer at run time. The buffer fields enable you to select the fields to be displayed.

Choose the Dynamic Buffer/Temp Table button on the SportsPro main window to launch this module:



This module demonstrates a non-SmartObject approach to separating the user interface and business logic portions of an application. The initializeObject procedure runs a procedure on the AppServer partition that populates the list of database tables. Choosing the Open Query button runs another procedure on the AppServer partition. That procedure populates a dynamic temp table and passes back the handle for that temp table.

Table 2–4 describes the major objects used in the Dynamic Buffer/Temp Table module.

Table 2–4: Components Of the Dynamic Buffer/Temp Table Module *(1 of 2)*

Object Or File Name	Object Type	Function
`wquery.w`	SmartWindow	Serves as a container for the other objects in the module.
`gettablefields.p`	Procedure file	Runs during the initializeObject internal procedure to populate the selecttable selection list.
selecttable	Selection-list	Lists the available tables.
`getnewfields.p`	Procedure file	Runs when a table is selected to populate the selectfld selection list.
selectfld	Selection-list	Lists the available fields in the selected table.
btnadd	Button	Adds the field selected in the selectfld selection list to the selectfld2 selection list through the additem internal procedure.
btnremove	Button	Removes the selected field from the selectfld2 selection list through the removeitem internal procedure.
selectfld2	Selection-list	Lists the fields that will be displayed when the query runs.
btnopenqry	Button	Runs `getrecords.p` and receives the handle of the dynamic temp-table that procedure creates. It then runs the displayflds internal procedure to populate EDITOR–1.
`getrecords.p`	Procedure file	Creates a dynamic temp-table and populates it based on the fields listed in the selectfld2 selection list.
EDITOR–1	Editor	Displays a single record from the dynamic temp-table.
btnfirst	Button	Navigates to the first row of the result set.
btnnext	Button	Navigates to the next row of the result set.

Table 2–4: Components Of the Dynamic Buffer/Temp Table Module *(2 of 2)*

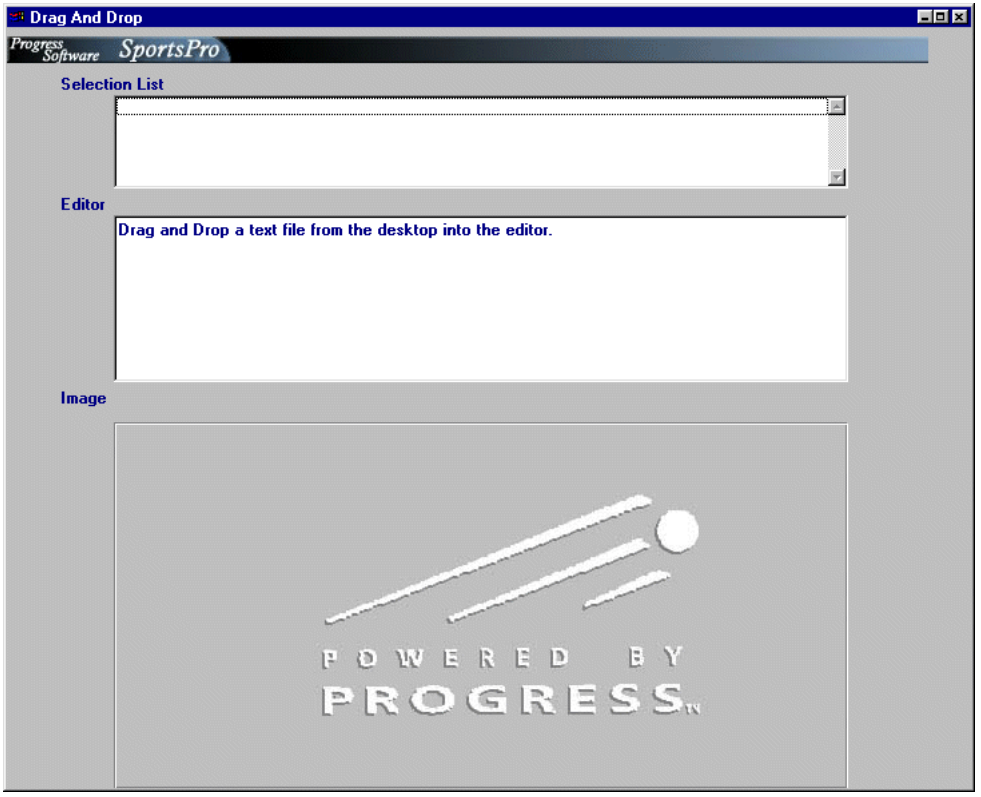
Object Or File Name	Object Type	Function
btnprev	Button	Navigates to the previous row of the result set.
btnlast	Button	Navigates to the last row of the result set.

For more information on building dynamic buffers and temp tables, see the chapters on temporary tables and on using dynamic widgets in the *Progress Programming Handbook*.

2.5.3 Drag and Drop Module

Progress can read the names of files that an end user drops onto certain widgets. With this functionality, the end user can populate widgets, like selection lists, with a list of files. Progress can also use the filenames to retrieve the files and open them in appropriate widgets. Using these techniques, you could save your users time by allowing them to load data into your program by dragging a filename from the Windows Explorer or Desktop onto a widget in your application.

Choose the Drag and Drop button on the SportsPro main window to launch this module:



Implementing this technique is simple. You set the DROP–TARGET attribute on the property sheet of a selection list or editor widget and code the trigger event. For images, the DROP–TARGET attribute is set on the frame that holds the image. When a file is dropped on an object with the DROP-TARGET attribute enabled, the DROP–FILE–NOTIFY event fires.

NOTE: If you cannot click on an object, it is probably behind another object in the window's layout. Use the Layout menu to move the top object back, or use the List Objects dialog box to access the bottom object.

Table 2–5 describes the major objects used in the Drag and Drop module.

Table 2–5: Components Of the Drag and Drop Module

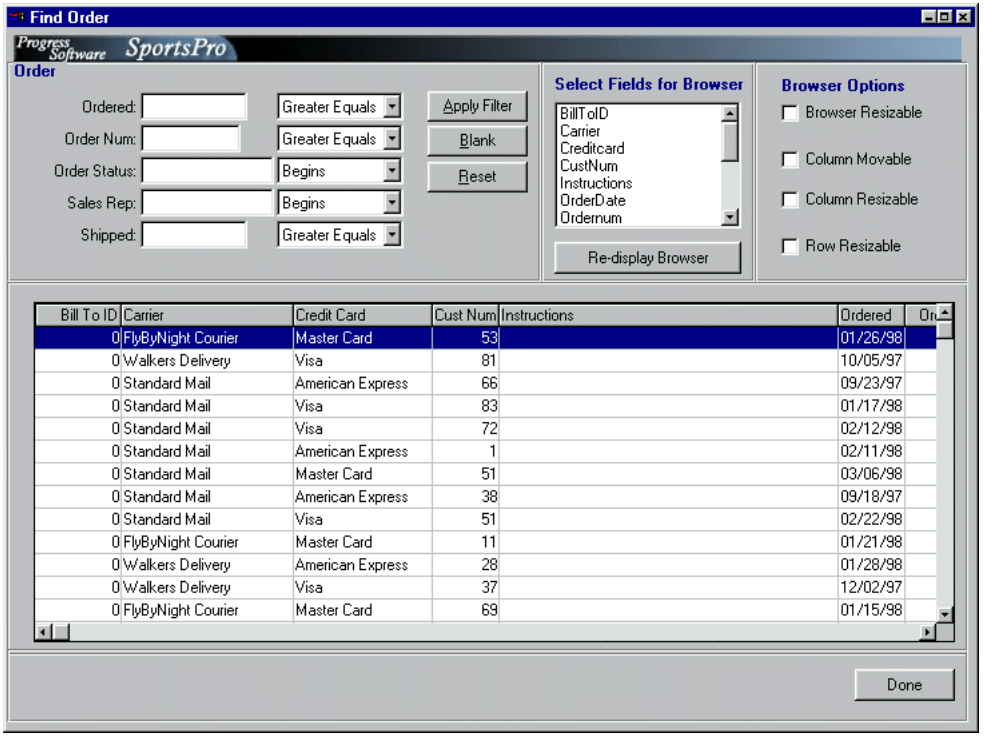
Object Or File Name	Object Type	Function
wdragdrop.w	SmartWindow	Serves as a container for the other objects in the module.
fMain	SmartFrame	Acts as the drop-target for image files. It reads the filename and then displays the image.
SELECT–1	Selection-list	Acts as the drop-target for filenames. It reads the filenames and then adds them to the list of selections.
EDITOR–1	Editor	Acts as the drop-target for text files. It reads the filename and then opens the file in the editor.

For more information on how drag and drop works, see the entries for the DROP–TARGET attribute, the END–FILE–DROP( ) method, the GET–DROPPED–FILE( ) method, the DROP–FILE–NOTIFY event, and the NUM–DROPPED–FILES attribute in the *Progress Language Reference*.

2.5.4 Resizable Dynamic Browser

Progress Version 9 introduced several new attributes for data browsers and viewers. This module gives you the opportunity to see what some of these new attributes do. The attributes demonstrated in this module allow an end user to change the way the data is displayed in the browser. The changes occur only in the user interface; they do not change field widths or column positions in the data source.

Choose the Resizable Dynamic Browser button on the SportsPro main window to launch this module:



In a normal browser, these attributes are set on the Advanced Properties dialog box in the AppBuilder at design time. Because the browser is dynamic, the attributes have to be set programmatically. The browser also uses the dynamic filters supplied as part of the ADM2.

Table 2–6 describes the major objects used in the Resizable Dynamic Browser module.

Table 2–6: Components Of the Resizable Dynamic Browser Module *(1 of 2)*

Object Or File Name	Object Type	Function
`wfndord.w`	SmartWindow	Serves as a container for the other objects in the module.
`dorder`	SmartDataObject	Queries the Order table to retrieve all order records for the query. The module accepts the handle for the instance of `dorder` in the SportsPro main window. **NOTE:** Because this module uses a handle passed in from the main window, you must run the main window and launch it from there.
`getnewfields.p`	Procedure file	Runs during the initializeObject internal procedure to populate the seldspflds selection list.
`dynfilter`	SmartFilter	Enables the user to dynamically filter the records shown in the browser.
seldspflds	Selection-list	Lists the fields in the Order table that can be displayed in the browser.
BUTTON-2	Button	Creates a dynamic browser instance from the master file, `adm2\dynbrowser.w`, using information from the filter and selection list.
dynbrowser	SmartDataBrowser	Enables the user to browse a result set created dynamically from `dorder` and information from the filter and selection list.
tglbrowserresizable	Toggle-box	Sets the RESIZABLE and SELECTABLE attributes of the browser.
tglcolmovable	Toggle-box	Sets the COLUMN–MOVABLE attribute of the browser.

Table 2–6: Components Of the Resizable Dynamic Browser Module *(2 of 2)*

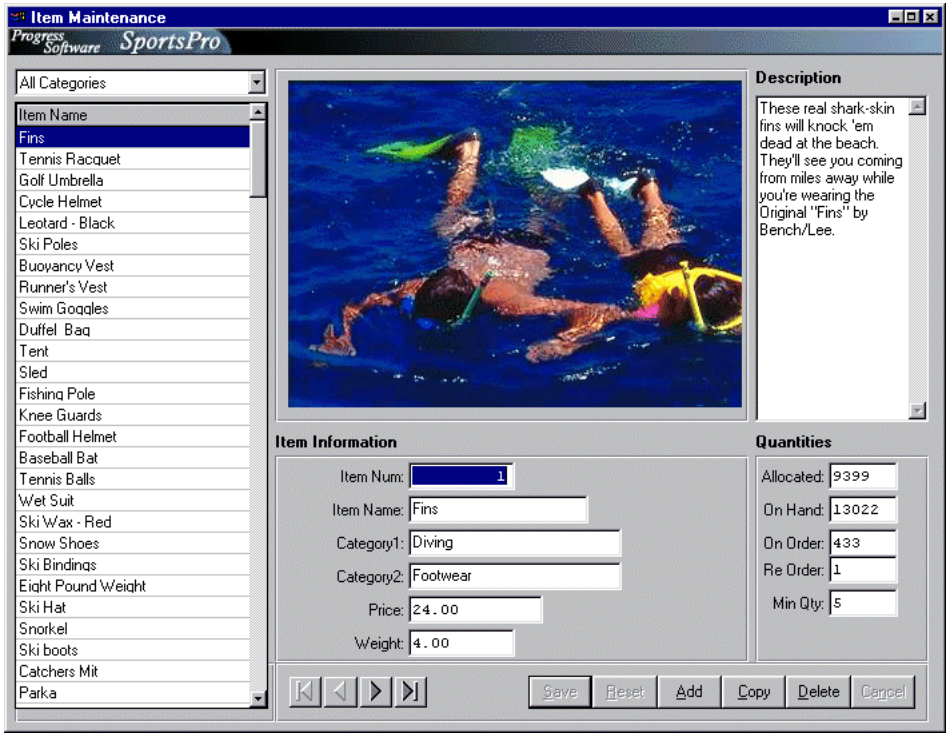
Object Or File Name	Object Type	Function
tglcolumnresizable	Toggle-box	Sets the COLUMN–RESIZABLE attribute of the browser.
tglrowresize	Toggle-box	Sets the ROW–RESIZABLE attribute of the browser.

For more information on resizable data browsers and viewers, see the chapters on using the browse widget and on direct manipulation in the *Progress Programming Handbook*.

2.5.5 SmartDataObject Module

One of the SmartObjects that Progress uses to support distributed applications is the SmartDataObject (SDO). The SDO manages the query, validation, and transaction logic in one object. One of its important features is that it can run locally, or be split between the client and AppServer in a distributed environment. Separating user interface from business logic is one of the keys to creating distributable applications. The SDO also provides an open interface that can function in an Open 4GL environment with non-Progress objects.

Choose the SmartDataObject button on the SportsPro main window to launch this module:



The SDO in this module queries the database and sends the records to a SmartDataBrowser and a SmartDataViewer. The SmartDataBrowser and SmartDataViewer are thin objects. They display data and accept changes to fields, but do not contain data update or validation logic. The SDO carries out the update and validation tasks on the server side.

Table 2–7 describes the major objects used in the SmartDataObject module.

Table 2–7: Components Of the SmartDataObject Module *(1 of 2)*

Object Or File Name	Object Type	Function
`wmntitem.w`	SmartWindow	Serves as a container for the other objects in the module.
`ditem`	SmartDataObject	Queries the Item table to retrieve item records.

Table 2–7: Components Of the SmartDataObject Module *(2 of 2)*

Object Or File Name	Object Type	Function
`getcategory.p`	Procedure file	Runs during the initializeObject internal procedure to populate the seldspflds selection list.
`bitem`	SmartDataBrowser	Enables the user to select an item record.
lcategory1	Combo-box	Filters the items shown in `bitem` to those matching the selected category.
`vitem`	SmartDataViewer	Displays detailed information for the item currently selected in `bitem`.
pnavico	SmartPanel (navigation)	Controls navigation for the result set from `ditem`.
pupdsav	SmartPanel (update)	Controls record update functions on the result set from `ditem`.

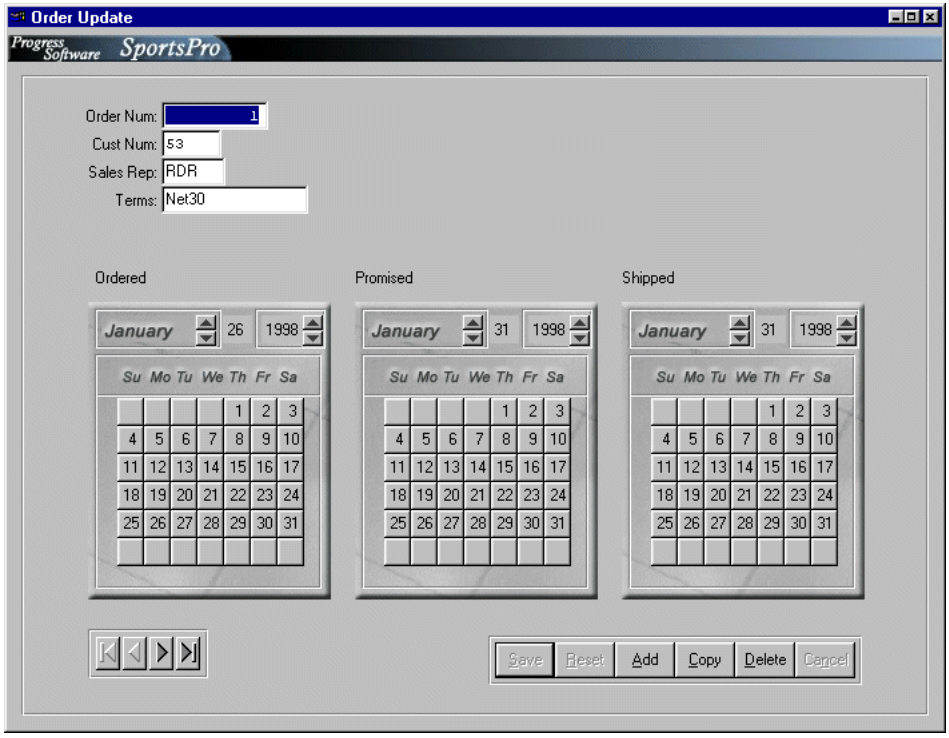
NOTE: The SmartDataObject module also runs when you choose the View→ Item Maintenance menu option in the SportsPro Order Tracking System.

For more information on SmartDataObjects, see the chapter on SmartObjects in the *Progress ADM 2 Guide*.

2.5.6 SmartDataField Module

The SmartDataField gives you a method to customize how particular pieces of data are displayed. You might use the SmartDataField as a calculator to update an integer field, or as an editor with cut, copy, and paste buttons to update a character field. In this module, a SmartDataField displays a date field as a calendar. Choosing a new date in the calendar changes the value in the field.

Choose the SmartDataField button on the SportsPro main window to launch this module:



SmartDataFields work with SmartDataViewers. You create a master for the SmartDataField that you want to use. Then you drop an instance of it onto an appropriate field in a SmartDataViewer to create the necessary links.

Table 2–8 describes the major objects used in the SmartDataField module.

Table 2–8: Components Of the SmartDataField Module *(1 of 2)*

Object Or File Name	Object Type	Function
wordupd.w	SmartWindow	Serves as a container for the other objects in the module.
dorder	SmartDataObject	Queries the Order table to retrieve all order records.
vorder2	SmartDataViewer	Displays the contents of several fields of the selected order record.

Table 2–8: Components Of the SmartDataField Module *(2 of 2)*

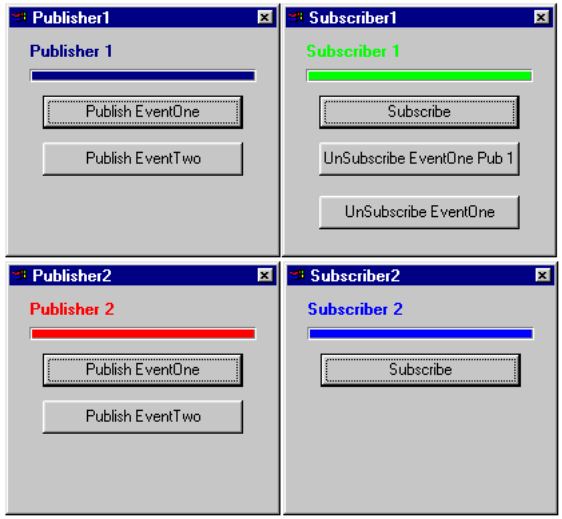
Object Or File Name	Object Type	Function
`sdfcal`	SmartDataField	Displays the value of the OrderDate, PromiseDate, and ShipDate fields in the selected record. The SmartDataFields also provide a more convenient method for changing the values than a standard fill-in field.
pnavico	SmartPanel (navigation)	Controls navigation for the result set from `dorder`.
pupdsav	SmartPanel (update)	Controls record update functions on the result set from `dorder`.

For more information on SmartDataFields, see the chapter on SmartObjects in the *Progress ADM 2 Guide*.

2.5.7 Publish and Subscribe Module

The Publish and Subscribe module demonstrates the Progress named-event functionality. The module uses two event publishers, `publisher1.w` and `publisher2.w`, and two event subscribers, `subscriber1.w` and `subscriber2.w`. A driver procedure, `subpub.p`, starts the publisher procedures as persistent procedures. The driver then starts the subscriber procedures, passing to them the handles for the persistent publisher procedures.

Choose the Publish and Subscribe button on the SportsPro main window to launch this module:



The publish and subscribe functionality works like an e-mail alias. When users subscribe to an e-mail alias, they receive messages whenever messages are published to that alias. Other persistent procedures running in the same session can locate a procedure that generates an event of interest. The persistent procedures subscribe to that event by mapping the event to an internal procedure that is executed when the event occurs.

Table 2–9 describes the major objects used in the Publish and Subscribe module.

Table 2–9: Components Of the Publish and Subscribe Module *(1 of 2)*

Object Or File Name	Object Type	Function
`suppub.p`	Procedure file	Runs the four windows as persistent procedures and passes the handles for the publishers to the subscribers.
`publisher1.w`	SmartWindow	Serves as a container for the objects in it.
BUTTON–1	Button	Publishes a named event, “EventOne,” which causes the EventOne internal procedure to run in any subscriber for which it currently has a handle. It reports the number of subscribers to the event from the replies it receives.

Table 2–9: Components Of the Publish and Subscribe Module *(2 of 2)*

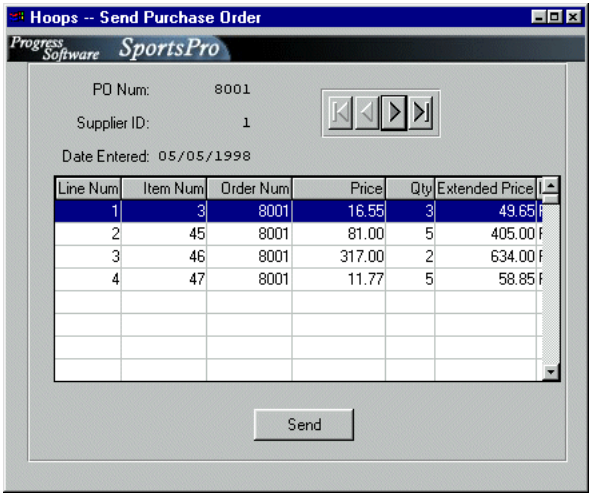
Object Or File Name	Object Type	Function
BUTTON–2	Button	Publishes a named event, “EventTwo,” which causes the EventTwo internal procedure to run in any subscriber for which it currently has a handle.
`publisher2.w`	SmartWindow	Serves as a container for the objects in it.
BUTTON–1	Button	Publishes a named event, “EventOne,” which causes the EventOne internal procedure to run in any subscriber for which it currently has a handle. It reports the number of subscribers to the event from the replies it receives.
BUTTON–2	Button	Publishes a named event, “EventTwo,” which causes the EventTwo internal procedure to run in any subscriber for which it currently has a handle.
subscriber1.w	SmartWindow	Serves as a container for the objects in it.
BUTTON–3	Button	Subscribes to both named events in `publisher1.w` and to “EventOne” in `publisher2.w`.
BUTTON–4	Button	Unsubscribes from “EventOne” in `publisher1.w`.
BUTTON–5	Button	Unsubscribes from “EventOne” in both publishers.
subscriber2.w	SmartWindow	Serves as a container for the objects in it.
BUTTON–6	Button	Subscribes to “EventTwo” in both publishers and to “EventOne” in `publisher1.w`.

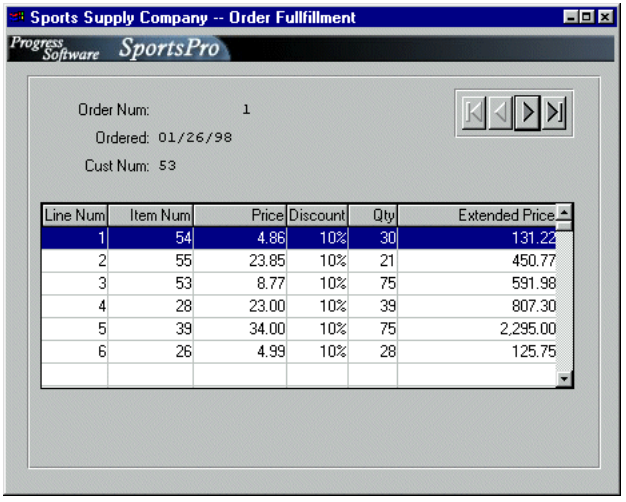
For more information on publishing and subscribing to named events, see the chapter on named events in the *Progress Programming Handbook*.

2.5.8 SmartB2B and SonicMQ Module

This module demonstrates the use of several new SmartObjects to link different businesses through a Publish and Subscribe model. You might also do this using a Point-to-Point model. A store can transmit its orders to its supplier as XML documents in a previously agreed upon format. The supplier can receive the XML document and translate the data into a form compatible with its own applications.

Choose the SmartB2B and SonicMQ button on the SportsPro main window. A window opens explaining the environment that you need to set up to run this module. Follow the instructions to launch the module:





Each of the applications that make up this module contains a SmartBusinessObject (SBO) that organizes the data streams of several SmartDataObjects. A SmartB2BObject takes the data from the SBO and runs it through a mapping file generated by the Progress XML Mapping Tool. The XML message is then sent out by a SmartProducer and transmitted using SonicMQ. The file is received at the other end by a SmartConsumer and passed to its SmartB2BObject to be decoded with another XML mapping file.

Table 2–10 describes the major objects used in the SmartB2B and SonicMQ module.

Table 2–10: Components Of the SmartB2B and SonicMQ Module *(1 of 3)*

Object Or File Name	Object Type	Function
gb2bintro.w	SmartDialog	Introduces the requirements for running this module. If you choose OK, wstore.w runs.
wstore.w	SmartWindow	Serves as a container for the other objects in the retailer's application.
sbopo	SmartBusinessObject	Organizes the data streams of dcust, ditem, dpo, and dpoline into a single logical data stream.
dcust	SmartDataObject	Queries the Customer table to retrieve all customer records, and supplies them to sbopo.
ditem	SmartDataObject	Queries the Item table to retrieve all item records, and supplies them to sbopo.
dpo	SmartDataObject	Queries the PurchaseOrder table to retrieve all purchase order records, and supplies them to sbopo.
dpoline	SmartDataObject	Queries the POLine table to retrieve all purchase order line records, and supplies them to sbopo.
vpo	SmartDataViewer	Displays fields from sbopo.
dynbrowser	SmartDataBrowser	Displays details of the line items in the record selected from sbopo.

Table 2–10: Components Of the SmartB2B and SonicMQ Module *(2 of 3)*

Object Or File Name	Object Type	Function
pnavico	SmartPanel (navigation)	Controls navigation within the result set from sbopo.
BUTTON–1	Button	Runs sendMessage( ) in sbbpo
sbbpo	SmartB2BObject	Using the predetermined schema, creates an XML message containing the data from the current record and passes it to dynproducer.
dynproducer	SmartProducer	Sends the XML message to the SonicMQ broker for delivery to the remote procedure.
wsupplier.w	SmartWindow	Serves as a container for the other objects in the supplier's application. **NOTE:** Since this module demonstrates how separate applications can exchange data, you have to start this application in a separate Progress session.
dynconsumer	SmartConsumer	Receives the XML message from the SonicMQ broker and passes it to sbborder.
sbborder	SmartB2BObject	Reads the data from the XML message and uses the predetermined schema to transform the XML message into a record in sboord.
sboord	SmartBusinessObject	Organizes the data streams of dord and dordline into a single logical data stream.
dord	SmartDataObject	Queries the Order table to retrieve all order records, and supplies them to sboord.
dordline	SmartDataObject	Queries the OrderLine table to retrieve all order line records, and supplies them to sboord.
vord	SmartDataViewer	Displays fields from sboord.

Table 2–10: Components Of the SmartB2B and SonicMQ Module *(3 of 3)*

Object Or File Name	Object Type	Function
dynbrowser	SmartDataBrowser	Displays details of the line items in the record selected from sboord.
pnavico-2	SmartPanel (navigation)	Controls navigation within the result set from sboord.

For more information on using SmartB2B objects and integrating Progress with SonicMQ, see the chapters on SmartObjects and accessing SonicMQ messaging from the Progress 4GL in *Progress Version 9 Product Update Bulletin*.

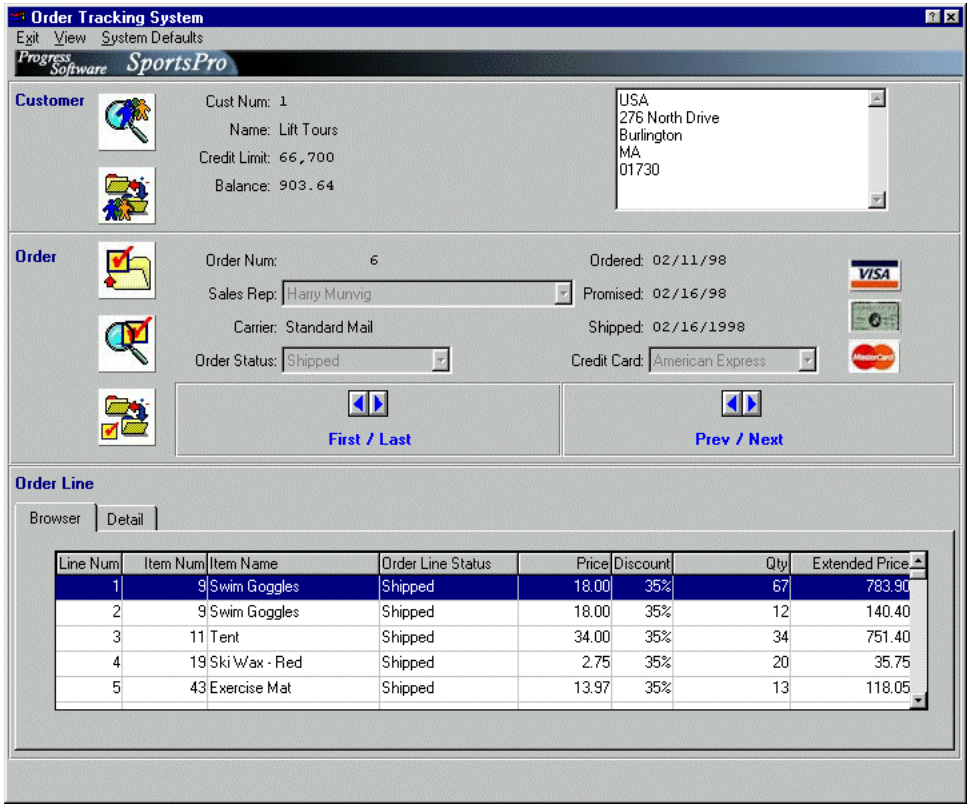
The SportsPro sample application modules that you examined are useful starting points. In them, you can see how to make better use of Progress' capabilities in your applications. To see how to integrate separate modules into a functioning application, you can now look at the last part of the sample application, the Order Tracking System. The Order Tracking System integrates most of the modules you have just examined and several others into a finished application.

2.6 Working With the Order Tracking System

The Order Tracking System combines modules to answer the common business functions needed in an order tracking application. Using it, you can add, search for, and maintain particular customers and orders. You can also maintain the product catalog and warehouse inventory records.

To begin examining the Order Tracking System, choose the Order Tracking button from the SportsPro sample application's main window.

The main window of the Order Tracking System opens:



NOTE: You might encounter problems with the application screens if you have your display set to large fonts and a resolution of 800x600. As a rule, if you use large fonts, you should use a higher resolution setting, such as 1024x768. In general, you can use large or small fonts with a higher resolution setting. If you do encounter problems with the application screens, try changing your display settings.

The main window is a SmartWindow that is divided into three parts:

- Customer section
- Order section
- Order Line section

Table 2–11 describes the major objects used in the main window of the Order Tracking System.

Table 2–11: Components Of the Order Tracking System Main Window *(1 of 2)*

Object Or File Name	Object Type	Function
worder.w	SmartWindow	Serves as a container for the other objects in the module.
m_Item_Maintenance	Menu-item	Launches wmntitem.w, which is described in Table 2–7.
m_Inventory_Trans	Menu-item	Launches winvtrans.w. The Inventory Transfer section is described below in Table 2–15.
m_Set_Application _Language	Menu-item	Launches wlang.w. The Set Application Language section is described below in Table 2–16.
dcust	SmartDataObject	Queries the Customer table to retrieve all customer records for the selected query.
vcust	SmartDataViewer	Displays the contents of several fields of the selected customer record.
bfndcust	Button	Launches wfndcust.w, passing it the handle of dcust. Table 2–3 describes wfndcust.w in detail.
bmntcust	Button	Launches wmntcust.w, passing it the handle of dcust. The Customer Maintenance section is described below in Table 2–12.
dorder	SmartDataObject	Queries the Order table to retrieve all order records for the selected query.
dordlne	SmartDataObject	Queries the OrderLine table to retrieve all order line records for the selected query.
vorder	SmartDataViewer	Displays the contents of several fields from the Order and Salesrep tables for the selected order record.

Table 2–11: Components Of the Order Tracking System Main Window *(2 of 2)*

Object Or File Name	Object Type	Function
dsalesrep	SmartDataObject	Queries the Salesrep table to retrieve all sales rep records for the selected query.
dynselect	SmartSelect	Allows the user to navigate through the records in dsalesrep.
CtrlFrame	OCX	Provides navigation control for the order records.
CtrlFrame-2	OCX	Provides navigation control for the order records.
bcreord	Button	Launches wmntord.w, passing it the handles of dcust, dorder, and dordlne. The Order Entry/Maintenance section is described below in Table 2–13.
bfndord	Button	Launches wfndord2.w, passing it the handle of dorder. The Find Order section is described below in Table 2–14.
bmntord	Button	Launches wmntord.w, passing it the handles of dcust, dorder, and dordlne. The Order Entry/Maintenance section is described below in Table 2–13.
folder	SmartFolder	Organizes the display of objects onto two logical pages, shown as the Browser and Detail tabs.
bordlne2	SmartDataBrowser	Allows the user to select an order line record.
vordlne	SmartDataViewer	Displays the contents of several fields of the selected order line record.
pnavico	SmartPanel (navigation)	Controls navigation for the result set from vordlne.
sports2000.hlp	Help Text File	Provide help text for the buttons in the main window of the sample application.

2.6.1 Customer Section

The Customer section of the Order Tracking System main window has a SmartDataViewer that displays customer information. From the main window:

- Choose the Find Customer button to open windows to search for customer information.
- Choose the Maintain Customer button to open windows to modify customer information.

Searching for Customers

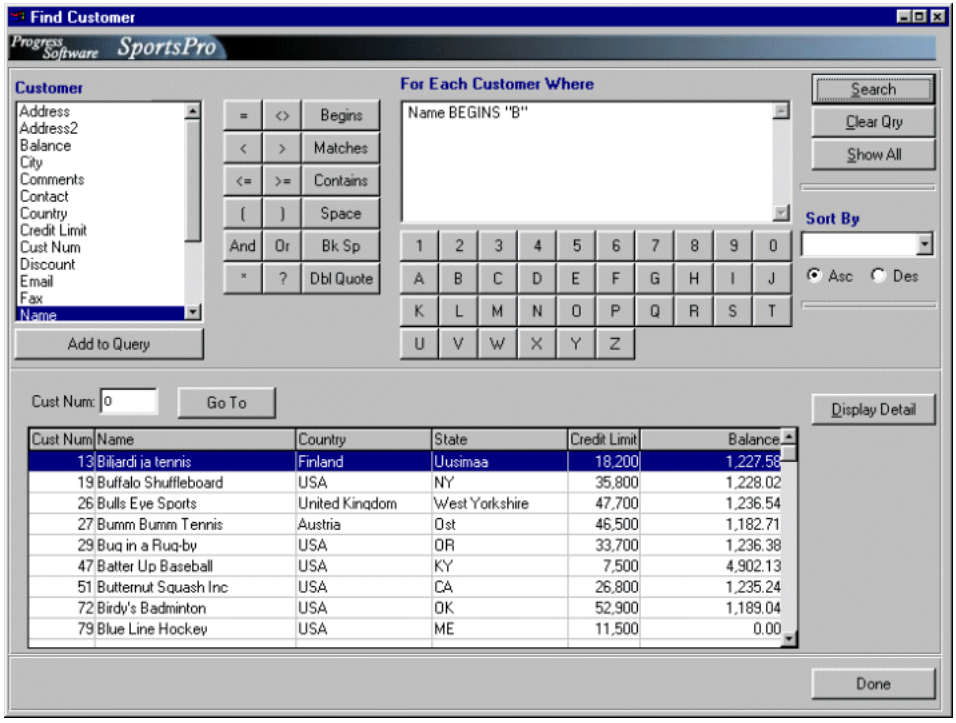
The Find Customer button in the Order Tracking System main window opens the Find Customer window. From this window, you can view information about all customers, or you can find customers and view information based on the search criteria you specify.

The Find Customer window is a nonmodal child window of the main window. As a result, you can move between the Find Customer window and the main window by clicking on them.

You can search the Sports2000 database dynamically by specifying query criteria at run time. Customers selected by the dynamic query appear in the browser.

Follow these steps to search for a customer:

1 ♦ Choose the Find Customer button. The Find Customer window opens:



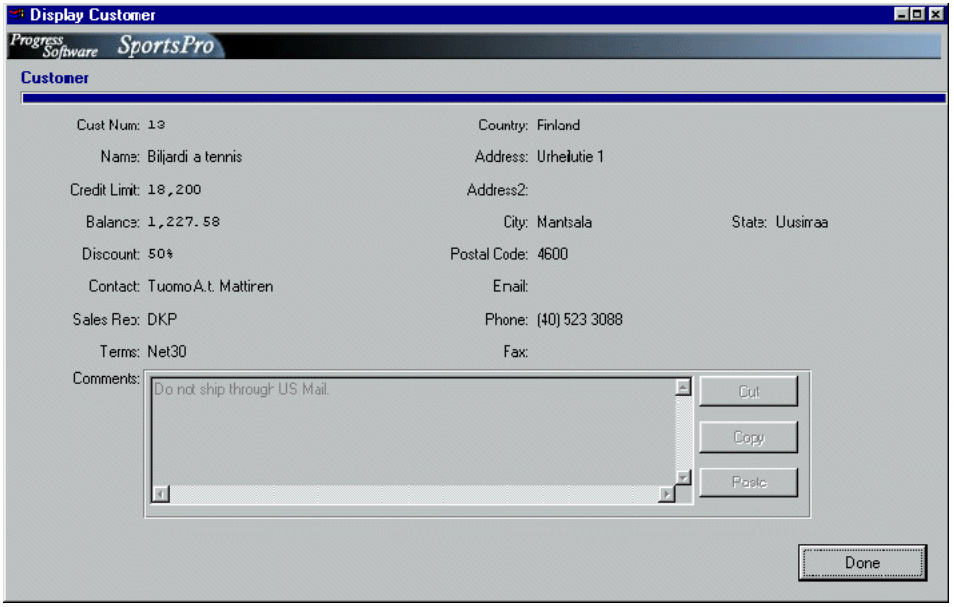
2 ♦ Using the Customer selection list and buttons in the upper section of the window, build the following query in the editor:

Name BEGINS "B"

3 ♦ Choose the Search button.

The SmartDataBrowser displays customer names that start with B. From this window, you view details about a selected customer. You can also use the Show All button to remove the query and to view all the customers.

4 ♦ Select a row in the browser and choose the Display Detail button. The Display Customer window opens:



The Display Customer window is a child window of the Find Customer window. The Display Customer window has a SmartDataViewer for displaying address and other information for the customer selected in the Find Customer window. To learn how to create this SmartDataViewer, see Chapter 4, "An Overview Of SmartObject Technology."

5 ♦ After viewing the information, close the Display Customer and Find Customer windows and return to the Order Tracking System main window.

NOTE: Table 2–3 describes the important elements of these windows in detail.

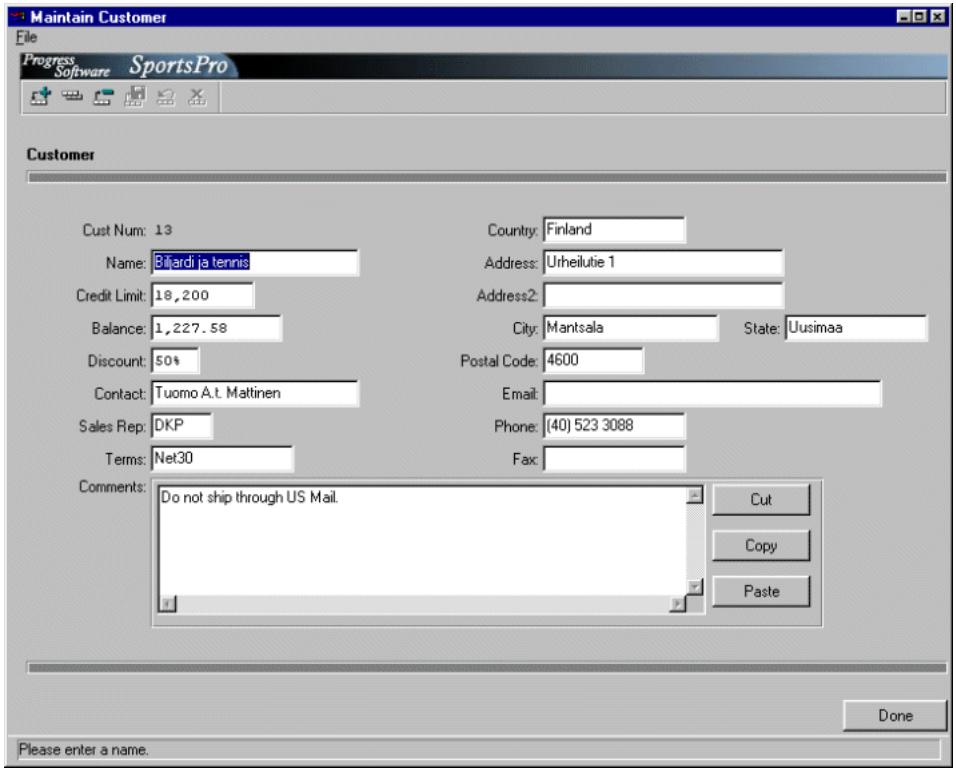
Modifying Customer Information

The Maintain Customer button in the Order Tracking System main window opens the Maintain Customer window. This window uses an instance of the same SmartDataViewer that you saw in the Find Customer window. In this instance, the SmartDataViewer is locally configured to allow you to change the record. This window also uses a SmartDataField on the Comments field to enable you to view and enter comments. There is also an instance of a dynamic SmartToolBar that you use to navigate, create, and maintain customer records.

The Maintain Customer window is a nonmodal child window of the main window. As a result, you can move between this window and the main window by clicking on them.

Follow these steps to modify customer information:

1 ♦ Choose the Maintain Customer button in the Order Tracking System main window. The Maintain Customer window opens:



You can enter and update changes in a customer's address, telephone number, contact, e-mail, and fax information using this window.

2 ♦ Close the Maintain Customer window and return to the Order Tracking System main window.

Table 2–12 describes the major objects in the Maintain Customer section of the Order Tracking System.

Table 2–12: Components Of the Maintain Customer Section

Object Or File Name	Object Type	Function
`wmntcust.w`	SmartWindow	Serves as a container for the objects in the Customer Maintenance section of this module.
`dcust`	SmartDataObject	Queries the Customer table to retrieve all customer records for the selected query. When this section is launched, it accepts the handle for the instance of `dorder` in the Order Tracking System main window. **NOTE:** This module uses a handle passed in from the main window. This module does not run properly unless launched from the main window.
`dyntoolbar`	SmartToolBar	Enables the user to add, copy, delete, or save the current customer record.
`vcust2`	SmartDataViewer	Displays the contents of several fields of the selected customer record.
`stxtfld`	SmartDataField	Displays the Comments field in `vcust2` in a Cut/Copy/Paste editor.

2.6.2 Order Section

The Order section of the Order Tracking System contains SmartObjects for querying, viewing, adding, and changing order information.

The buttons in this section launch modules for performing the following tasks:

- Place new orders for a customer
- View order information
- View, update, and delete orders

Placing New Orders

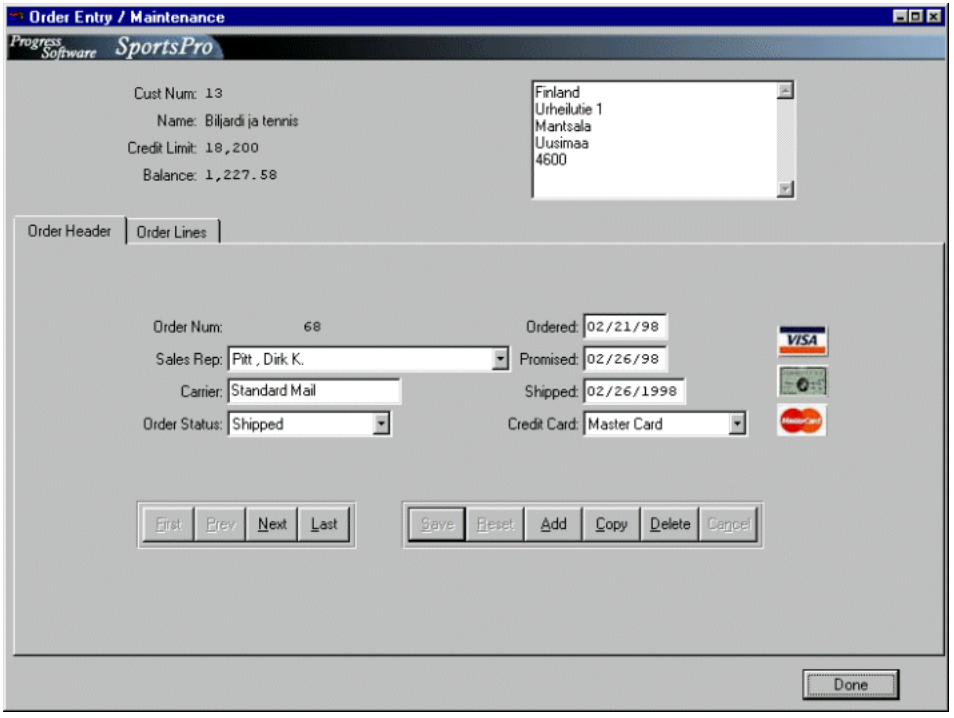
The Place New Order button in the Order section opens the Order Entry/Maintenance window to create new orders.

The Order Entry/Maintenance window is a nonmodal child window of the main window. As a result, you can move between the Order Entry/Maintenance window and the main window by clicking on them.

The Order/Entry Maintenance window has a SmartDataViewer for viewing order information and a SmartFolder with two tabs. A SmartFolder is a SmartObject that organizes the display of information into logical groupings, usually presenting the most frequently needed information on the first tab. In this instance, you can view different levels of detail about the customer's orders and the individual lines of an order by choosing different tabs.

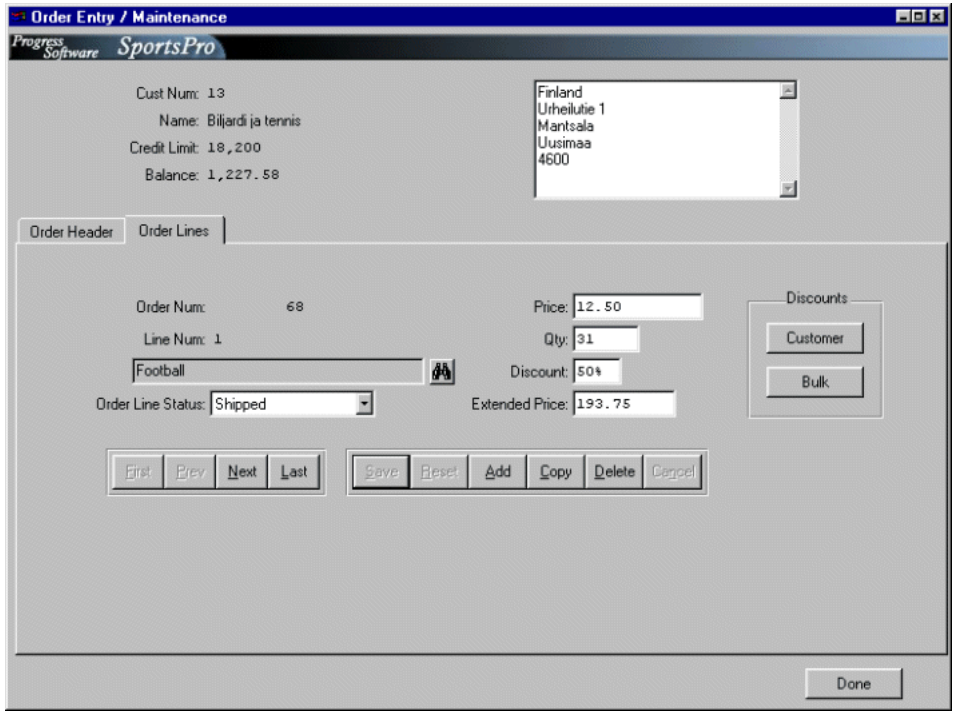
Follow these steps to view order information and place a new order:

1 ♦ Choose the Place New Order button. The Order Entry/Maintenance window opens:



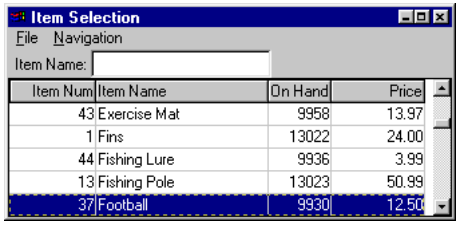
The Order Header tab contains a SmartDataViewer that displays information about the order currently shown in the Order section of the Order Tracking System main window. The Order Header tab also contains a Navigation panel and a record Update panel.

2 ♦ Choose the Order Lines tab to view the Order Lines information:



This tab contains a SmartDataViewer that displays information about the individual order lines in the current order.

3 ♦ Choose the browse button next to the Item field to open the Item Selection window:



The Item Selection window is a SmartSelect Browser window with a dynamic SmartDataBrowser for browsing item information.

4 ♦ Close the Item Selection and Order Entry/Maintenance windows, and return to the Order Tracking System main window.

Table 2–13 describes the major objects in the Order Entry/Maintenance section of the Order Tracking System.

Table 2–13: Components Of the Order Entry/Maintenance Section *(1 of 3)*

Object Or File Name	Object Type	Function
`wmntord`	SmartWindow	Serves as a container for the objects in the Order Entry/Maintenance section of this module.
`dcust`	SmartDataObject	Queries the Customer table to retrieve all customer records for the selected query. When this module is launched, it accepts the handle for the instance of `dcust` in the Order Tracking System main window. **NOTE:** This module uses a handle passed in from the main window. This module does not run properly unless launched from the main window.
`dorder`	SmartDataObject	Queries the Order table to retrieve all order records for the selected query. When this module is launched, it accepts the handle for the instance of `dorder` in the Order Tracking System main window. **NOTE:** This module uses a handle passed in from the main window. This module does not run properly unless launched from the main window.

Table 2–13: Components Of the Order Entry/Maintenance Section *(2 of 3)*

Object Or File Name	Object Type	Function
dordlne	SmartDataObject	Queries the OrderLine table to retrieve all order line records for the selected query. When this module is launched, it accepts the handle for the instance of dordlne in the Order Tracking System main window. **NOTE:** This module uses a handle passed in from the main window. This module does not run properly unless launched from the main window.
vcust	SmartDataViewer	Displays the contents of several fields of the selected customer record.
folder	SmartFolder	Organizes the display of objects onto two logical pages, shown as the Order Header and Order Lines tabs.
vorder	SmartDataViewer	Displays the contents of several fields from the Order and Salesrep tables for the selected order record.
dsalesrep	SmartDataObject	Queries the Salesrep table to retrieve all sales rep records for the selected query.
dynselect	SmartSelect	Allows the user to navigate through the records in dsalesrep.
pnavlbl	SmartPanel (navigation)	Allows the user to navigate through the current result set.
pupdsav	SmartPanel (update)	Allows the user to add, change, or delete records.
vordlne2	SmartDataViewer	Displays the contents of several fields of the selected order line record.
ditem	SmartDataObject	Queries the Item table to retrieve item records for vordlne2.
dynselect	SmartSelect	Allows the user to navigate through the records in ditem.

Table 2–13: Components Of the Order Entry/Maintenance Section *(3 of 3)*

Object Or File Name	Object Type	Function
btCustomerDiscount	Button	Runs `getcustdiscount.p`. Reads the returned value and then runs the calculatePrice internal procedure.
`getcustdiscount.p`	Procedure file	Returns the discount from the Customer table.
btBulkDiscount	Button	Runs `getbulkdiscount.p`. Reads the returned value and then runs the calculatePrice internal procedure.
`getbulkdiscount.p`	Procedure file	Returns the minimum quantities needed to qualify for a discount for each item.
`pnavlbl-2`	SmartPanel (navigation)	Allows the user to navigate a set of records in a SmartDataObject.
pupdsav-2	SmartPanel (update)	Allows the user to add, change, or delete records.

Finding Orders

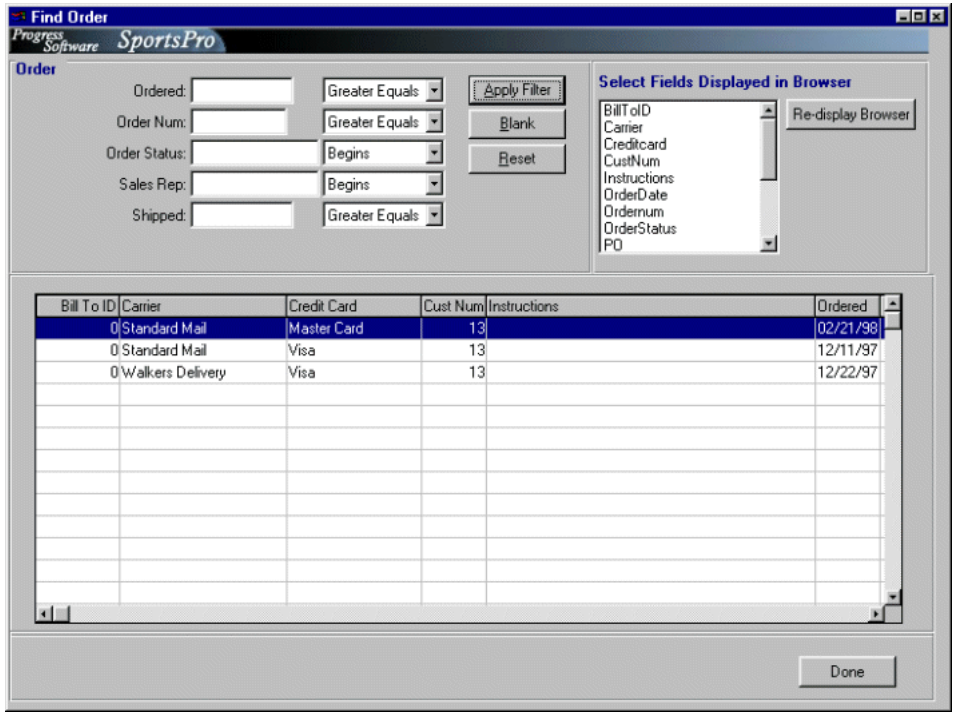
The Find Order button in the Order section opens the Find Order window to view order information.

The Find Order window is a nonmodal child window that is called from the main window. As a result, you can move between the Find Order window and the main window by clicking on them.

The Find Order window contains a SmartFilter and a dynamic browser. Use the SmartFilter to define criteria to search through the selected customer’s orders. You can define a search query based on the date ordered, the order number, the order status, the sales representative, and the shipped date. Use the selection list to control which fields are displayed in the dynamic browser.

Follow these steps to view order information using a dynamic query:

1 ♦ Choose the Find Order button. The Find Order window opens:



Initially, the window displays all available orders for the selected customer. You can use the Order filter section to define new search criteria to limit the orders that appear. You use the selection list, Selected Fields Displayed in Browser, to define which fields you want to have displayed in the dynamic browser.

2 ♦ Type **01/01/98** into the Shipped field and choose the Apply Filter button.

The browser now displays order information based on the shipped date you entered.

NOTE: The fields in the browser remain the same as those in the initial search.

3 ♦ From the Select Fields Displayed in Browser list, select the following fields for the browser:

- CustNum
- OrderDate
- Ordernum
- OrderStatus
- SalesRep
- ShipDate

NOTE: Use the **CTRL** key to select multiple fields.

4 ♦ Choose the Re-display Browser button to display the new fields in the dynamic browser.

The browser displays the new fields you selected.

5 ♦ Close the Find Customer window and return to the main window.

Table 2–14 describes the major objects in the Find Order section of the Order Tracking System.

Table 2–14: Components Of the Find Order Section *(1 of 2)*

Object Or File Name	Object Type	Function
`wfndord2.w`	SmartWindow	Serves as a container for the other objects in Find Order section of this module.
`dorder`	SmartDataObject	Queries the Order table to retrieve all order records for the selected query. When this module is launched, it accepts the handle for the instance of `dorder` in the Order Tracking System main window. **NOTE:** This module uses a handle passed in from the main window. This module does not run properly unless launched from the main window.

Table 2–14: Components Of the Find Order Section *(2 of 2)*

Object Or File Name	Object Type	Function
`getnewfields.p`	Procedure file	Runs during the initializeObject internal procedure to populate the seldspflds selection list.
`dynfilter`	SmartFilter	Enables the user to dynamically filter the records shown in the browser.
seldspflds	Selection-list	Lists the fields in the Order table that can be displayed in the browser.
BUTTON-2	Button	Creates a dynamic browser instance from the master file, `adm2\dynbrowser.w`, using information from the filter and selection list.
dynbrowser	SmartDataBrowser	Enables the user to browse a result set created dynamically from `dorder` and information from the filter and selection list.

Changing Orders

The Maintain Order button in the Order section opens the Order/Entry Maintenance window. This is the same module that you examined in the "Placing New Orders" section. There are two entry points for this module to allow for differences in the way orders are created and updated.

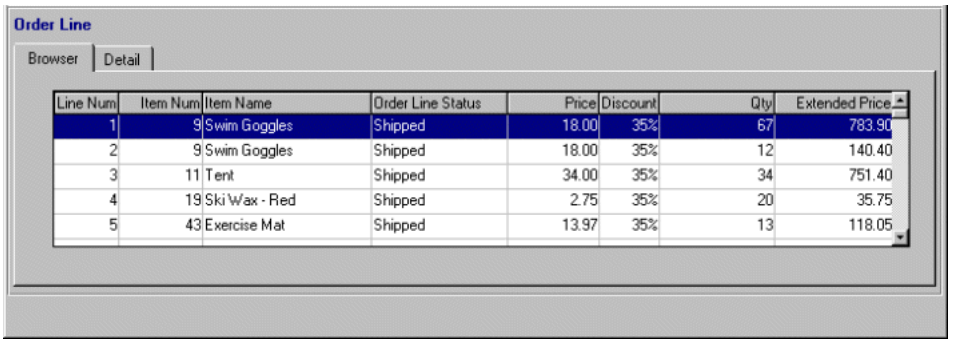
If your business rules include differences in how orders are entered and maintained, you would code custom triggers on these buttons to enforce those rules. For example, if only field agents could enter orders, you would code the Order Entry button to be disabled for anyone who was not a field agent.

2.6.3 Order Line Section

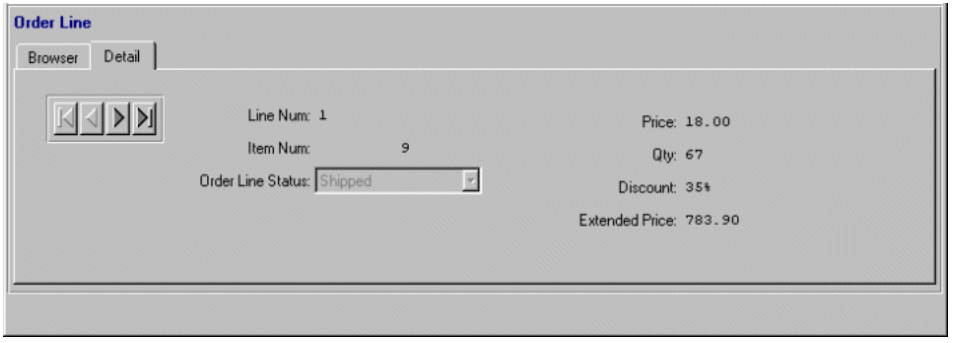
The Order Line section of the Order Tracking System main window contains a SmartFolder with two tabs. One tab has a SmartDataBrowser for viewing Order Lines. The other tab has a SmartDataViewer for viewing details about the Order Line. The display of the tabs in the SmartDataViewer is controlled using logical pages. For more information about logical pages, see the "Building the Order Line Section" section in Chapter 3, "Assembling the Order Tracking System."

Follow these steps to view the tabs in the SmartFolder:

1 ♦ Choose the Browser tab and select a particular line of the order:



2 ♦ Choose the Detail tab to view the details of just that line of the order:

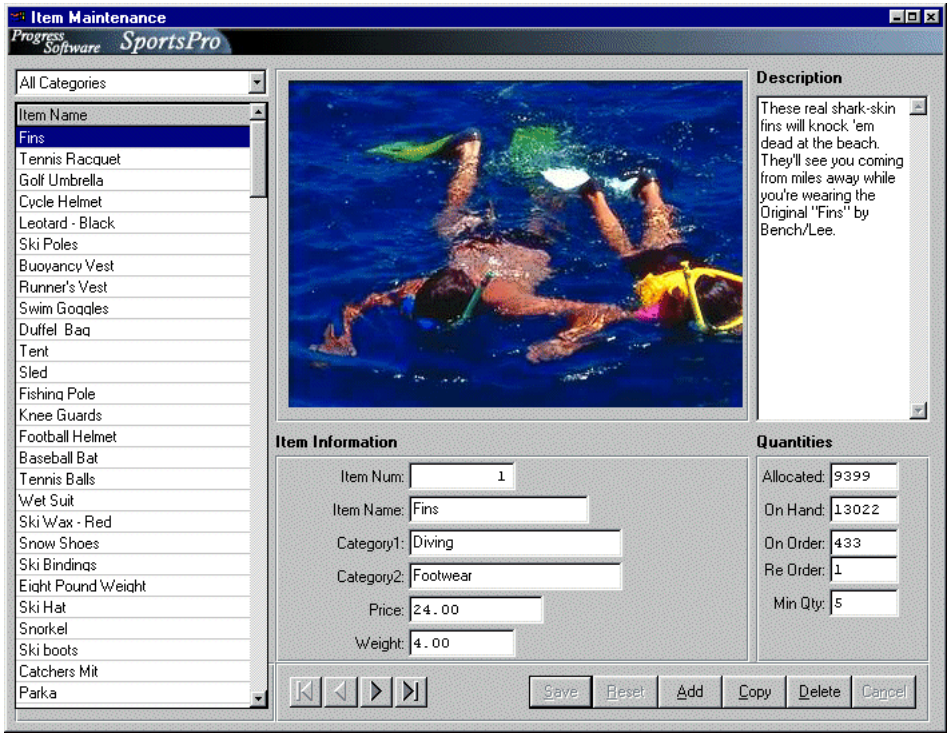


2.6.4 Menu Items

Less commonly used modules of the Order Tracking System can be launched through menu options. The Order Tracking System has menu options for maintaining the product catalog, examining warehouse records, and setting the language in which the application displays.

Follow these steps to view the modules available from the menu:

1 ♦ Choose View→ Item Maintenance. The Item Maintenance window opens:

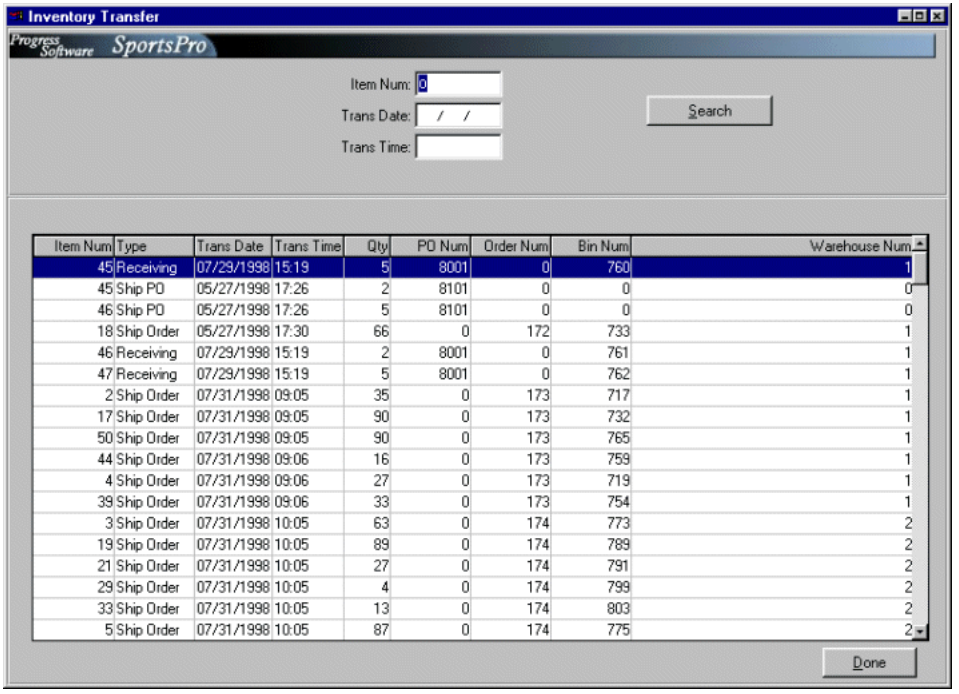


The Item Maintenance window is a SmartWindow with a SmartDataBrowser, a SmartDataViewer, a Navigation SmartPanel, and an Update SmartPanel.

NOTE: Table 2–7 describes the important elements of this window.

2 ♦ After examining this window, close it and return to the Order Tracking System main window.

3 ♦ Choose View→ Inventory Transfers. The Inventory Transfer dialog box opens:



Item Num	Type	Trans Date	Trans Time	Qty	PO Num	Order Num	Bin Num	Warehouse Num
45	Receiving	07/29/1998	15:19	5	8001	0	760	1
45	Ship PO	05/27/1998	17:26	2	8101	0	0	0
46	Ship PO	05/27/1998	17:26	5	8101	0	0	0
18	Ship Order	05/27/1998	17:30	66	0	172	733	1
46	Receiving	07/29/1998	15:19	2	8001	0	761	1
47	Receiving	07/29/1998	15:19	5	8001	0	762	1
2	Ship Order	07/31/1998	09:05	35	0	173	717	1
17	Ship Order	07/31/1998	09:05	90	0	173	732	1
50	Ship Order	07/31/1998	09:05	90	0	173	765	1
44	Ship Order	07/31/1998	09:06	16	0	173	759	1
4	Ship Order	07/31/1998	09:06	27	0	173	719	1
39	Ship Order	07/31/1998	09:06	33	0	173	754	1
3	Ship Order	07/31/1998	10:05	63	0	174	773	2
19	Ship Order	07/31/1998	10:05	89	0	174	789	2
21	Ship Order	07/31/1998	10:05	27	0	174	791	2
29	Ship Order	07/31/1998	10:05	4	0	174	799	2
33	Ship Order	07/31/1998	10:05	13	0	174	803	2
5	Ship Order	07/31/1998	10:05	87	0	174	775	2

The Inventory Transfer dialog box is a SmartDialog that is a child of the main window. The Inventory Transfer dialog box displays inventory records from the database. The fill-in fields enable you to specify search criteria to narrow the records shown in the SmartDataBrowser.

Table 2–15 describes the major objects in the Inventory Transfer section of the Order Tracking System.

Table 2–15: Components Of the Inventory Transfer Section *(1 of 2)*

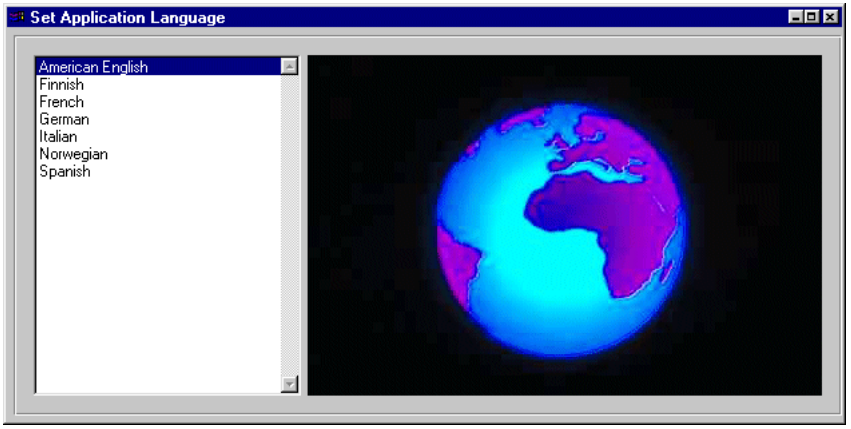
Object Or File Name	Object Type	Function
winvtrans.w	SmartWindow	Serves as a container for the objects in the Inventory Transfer section of this module.
`dinvtrans`	SmartDataObject	Queries the InventoryTrans table for inventory information.

Table 2–15: Components Of the Inventory Transfer Section *(2 of 2)*

Object Or File Name	Object Type	Function
`binvtrans`	SmartDataBrowser	Enables the user to select an inventory record.
btn-search	Button	Uses any values in the fill-in fields to replace the WHERE clause in `dinvtrans`.

4 ♦ Close the Inventory Transfer module and return to the main window.

5 ♦ Choose System Defaults→ Set Application Language. The Set Application Language window opens:



Progress applications can be translated into many languages using the Translation Manager tool. This module enables each user to choose among the available translations for the application.

Table 2–16 describes the major objects in the Set Application Language section of the Order Tracking System.

Table 2–16: Components Of the Set Application Language Section

Object Or File Name	Object Type	Function
`wlang.w`	SmartWindow	Serves as a container for the objects in the Set Application Language section.
SELECT-1	Selection-list	Lists the available languages in which the module can display. Selecting a new language sets it as the current language, if you have it installed on your machine.

6 ♦ Close all the open windows to exit from the SportsPro sample application.

You can leave the PRO*Tools palette open or close PRO*Tools by choosing the Close button in the upper-right corner of the PRO*Tools palette.

Now that you have worked with the sample application, you have a sense of how a user would interact with an application and how to build applications using the AppBuilder and the Progress SmartObject technology. For a better understanding of how to use the AppBuilder to assemble the sample application, see Chapter 3, "Assembling the Order Tracking System."

For an overview of SmartObject technology and information on fabricating your own SmartObjects, see Chapter 4, "An Overview Of SmartObject Technology."

3

Assembling the Order Tracking System

This chapter is a tutorial that leads you through assembling the main window of the SportsPro Order Tracking System sample application from the sample SmartObjects supplied by Progress. Assembling the main window can help you better understand SmartObject technology and how to use the AppBuilder to build an application.

The assembly process consists of creating a new SmartWindow to serve as a container for the other objects, adding the SmartObjects to the SmartWindow one at a time, and, as you do this, creating SmartLinks that allow the various SmartObjects to exchange information and instructions with each other in a standardized way.

This chapter has the following sections:

- Progress Advisor
- Tutorial Structure
- Assembling the Main Window
- Building the Customer Section
- Building the Order Section
- Building the Order Line Section

3.1 Progress Advisor

While assembling the SportsPro Order Tracking System, you receive helpful information and suggestions on creating SmartLinks from the Progress Advisor. SmartLinks provide a direct communication pathway between SmartObjects. The Progress Advisor is a dialog box that opens whenever you add a new SmartObject that could potentially serve as a source or target for a SmartLink.

The usefulness of the Progress Advisor's suggestions depends on the way you plan your application's development. The way to make the best use of the Advisor is to plan the application around a required workflow and to assemble the SmartObjects and other components in a way that reflects the workflow as closely as possible. For more information about SmartLinks and planning an application, see Chapter 4, "An Overview Of SmartObject Technology."

3.2 Tutorial Structure

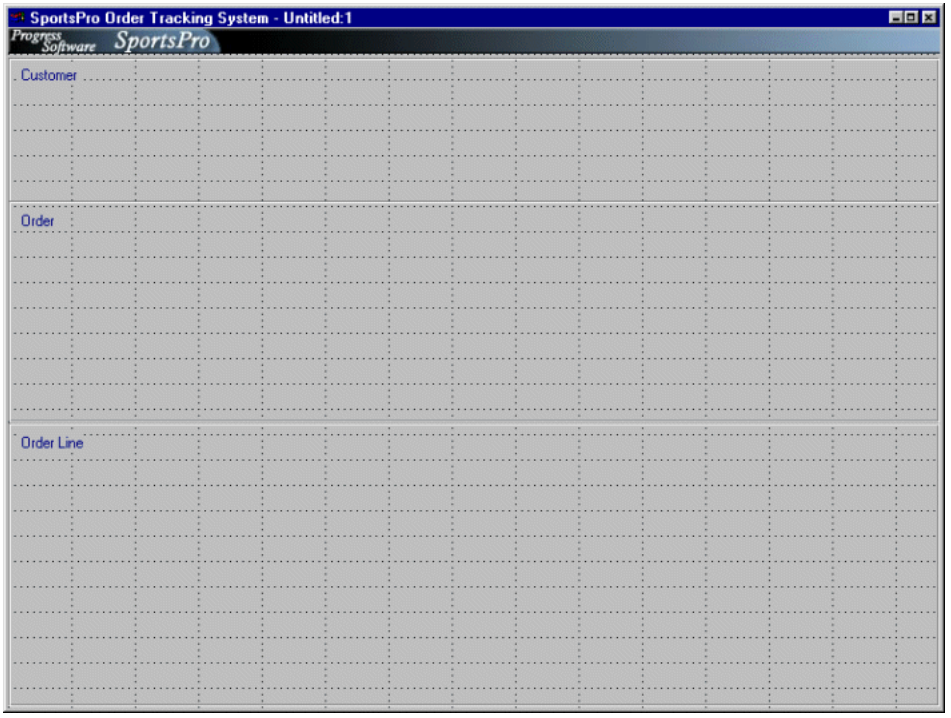
The tutorial is divided into sections that outline a related group of tasks. Each section begins by showing you a screen shot of what your application should look like when you complete that part of the tutorial. The section then gives a list of tasks that you should be familiar with in order to complete the section.

If you already know how to complete these tasks, you should try the exercise on your own. Below the task list is a condensed set of steps to follow. These steps provide you with basic information you need, such as the dimensions of the objects you should build. If you want to review a particular procedure in detail, the cross-references under each step enable you to find that procedure in the tutorial that follows.

The final part of each section is a detailed tutorial. This tutorial leads you through the exercise for each section in detail appropriate to a developer with some familiarity with graphical user interface (GUI) design environments.

3.3 Assembling the Main Window

The first stage of creating the SportsPro Order Tracking System module is building the main window and dividing it into sections. When you are finished with this stage, your window should look something like the following:



3.3.1 Task List and Short Procedure

To reach this point, you need to know how to do the following tasks:

1. Access the New dialog box.
2. Access an object's Property Sheet.
3. Use your mouse to move and resize an object.
4. Add an image to a frame.
5. Create and resize a rectangle.

6. Create text labels.

7. Access the List Objects dialog box.

8. Save your file.

NOTE: During application assembly, the main window is also referred to as the workspace or design window.

The following short procedure outlines how to complete these tasks:

1 ♦ Create a new SmartWindow.

To review this procedure, see the "Detailed Main Window Procedure" section.

2 ♦ Set it to be 147.00 units wide and 25.50 units high.

To review this procedure, see Step 3 in the "Detailed Main Window Procedure" section.

3 ♦ Resize the default frame.

To review this procedure, see Step 5 in the "Detailed Main Window Procedure" section.

4 ♦ Add the application banner.

To review this procedure, see the "Defining Sections Of the Main Window" section.

5 ♦ Divide the window into three sections with rectangles with the following properties:

Column	Row	Width	Height
1.00	2.1	147	5.48
1.00	7.48	147	8.33
1.00	15.81	147	10.67

To review this procedure, see the "Outlining the Customer, Order, and Order Line Sections" section.

6 ♦ Add dark blue text labels naming the three sections: Customer, Order, Order Line.

To review this procedure, see the "Adding Text Labels To the Sections" section.

7 ♦ Make “SportsPro Order Tracking System” the title of your window,

To review this procedure, see the “Naming and Saving the New SmartWindow” section

8 ♦ Save your window in a file named `wordtrk.w`.

To review this procedure, see Step 5 in the “Naming and Saving the New SmartWindow” section.

3.3.2 Detailed Main Window Procedure

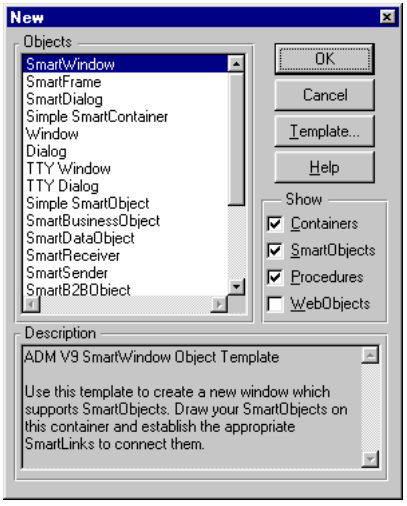
The following sections lead you through assembling your main window in more detail.

Creating a SmartWindow

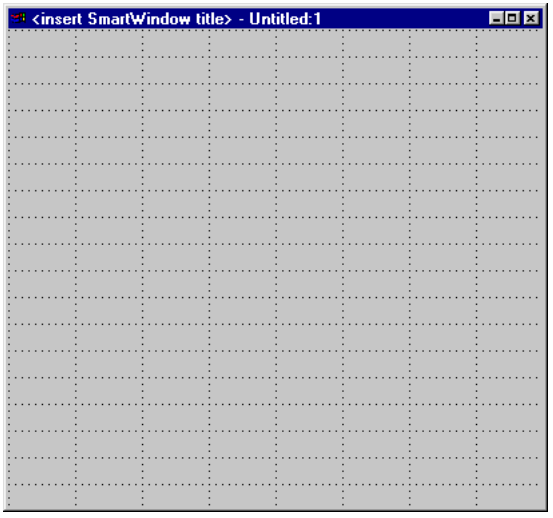
The first step in assembling the main window is to create a new SmartWindow to serve as a container for other SmartObjects.

Follow these steps to create a new SmartWindow:

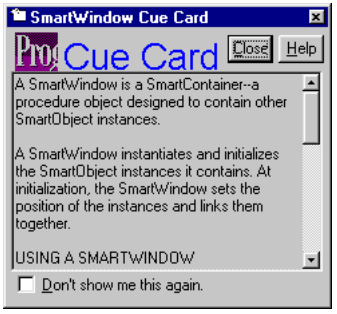
1 ♦ Open the New dialog box and choose a SmartWindow:



A new, untitled SmartWindow opens:



A SmartWindow Cue Card also opens:

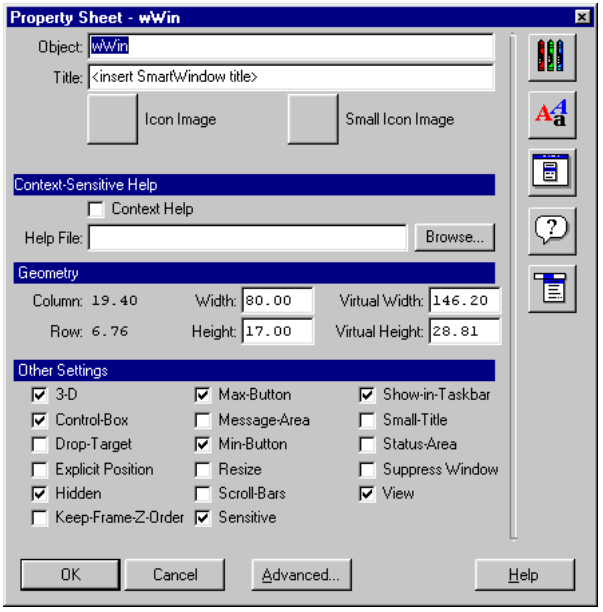


Each time you create a new SmartObject, the AppBuilder displays a Cue Card for the type of SmartObject you are creating. The Cue Card provides useful information including an explanation of how the SmartObject works and key steps to follow while creating that SmartObject.You do not need to use the Cue Cards for this tutorial.

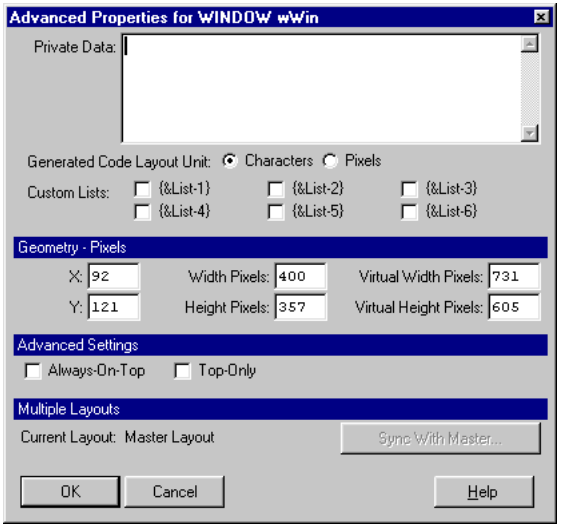
Remember, when you use the AppBuilder to develop an application, Progress Cue Cards are available to assist you. If you close the Progress Cue Cards, you can always open them again using the Help→ Cue Card option from the AppBuilder menu bar.

2 ♦ Choose the Object Properties icon  in the AppBuilder toolbar.

The property sheet for the new SmartWindow opens. A property sheet allows you to specify various properties such as the dimensions and position of a widget, a visual element of a graphical user interface. A basic property sheet of a widget allows you to specify the column, row, height, and width in character units, as shown in the following figure:

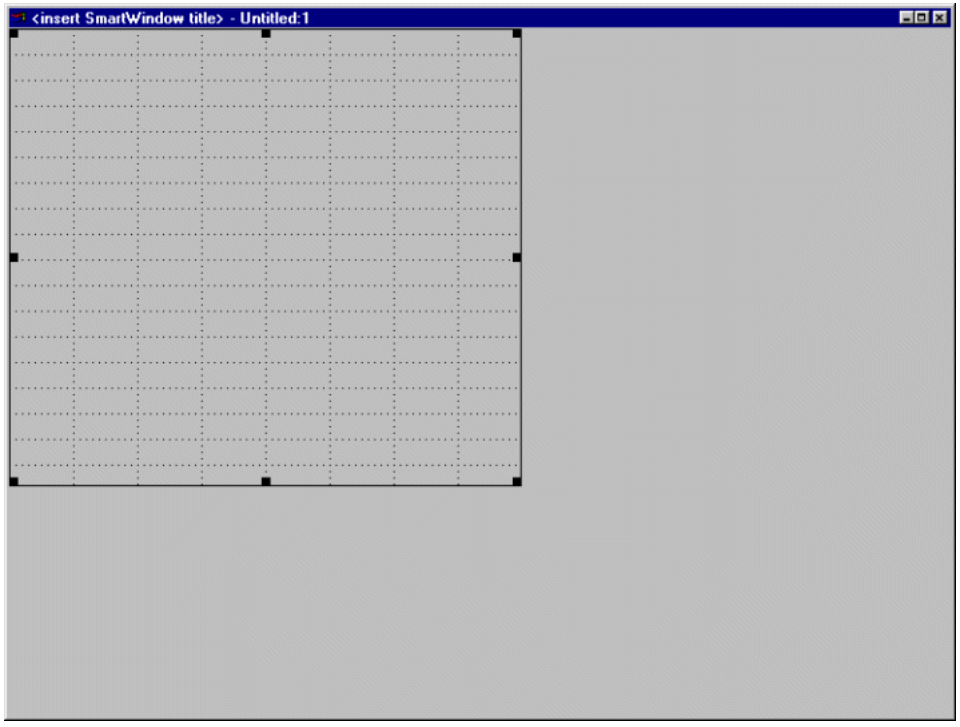


The Advanced button on the Property Sheet opens the Advanced Properties dialog box. A widget's Advanced property sheet allows you to specify the x and y coordinates of the widget and its height and width in pixels. In both the Basic and Advanced property sheets, the widget's coordinates are relative to its parent. The coordinates of a window are relative to the screen:



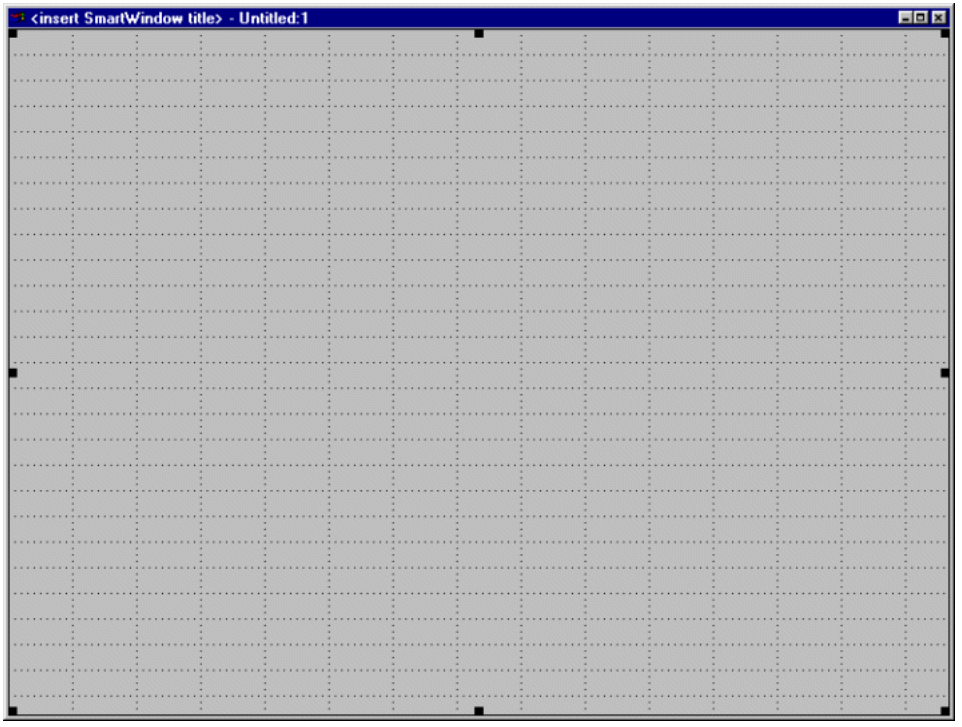
3 ♦ In the Geometry section of the Basic property sheet, type **147.00** for the Width and **25.50** for the Height. These dimensions resize the window so that it is large enough to hold the objects you are going to add.

4 ♦ Choose OK in the property sheet. The main window reopens with its new dimensions:



Notice that the default frame retained the original dimensions of the SmartWindow. Because SmartContainers can contain more than one frame, their default frames do not automatically resize when you expand the SmartContainer.The frames in design windows for other types of SmartObjects do resize automatically with the window.

5 ♦ Use the frame's handles to make the frame fill the entire SmartWindow:



You are now ready to define the sections of the main window. When you ran the sample application in Chapter 2, "Working With the Sample Application," you saw that the Order Tracking System main window is divided into these sections:

- Banner
- Customer
- Order
- Order Line

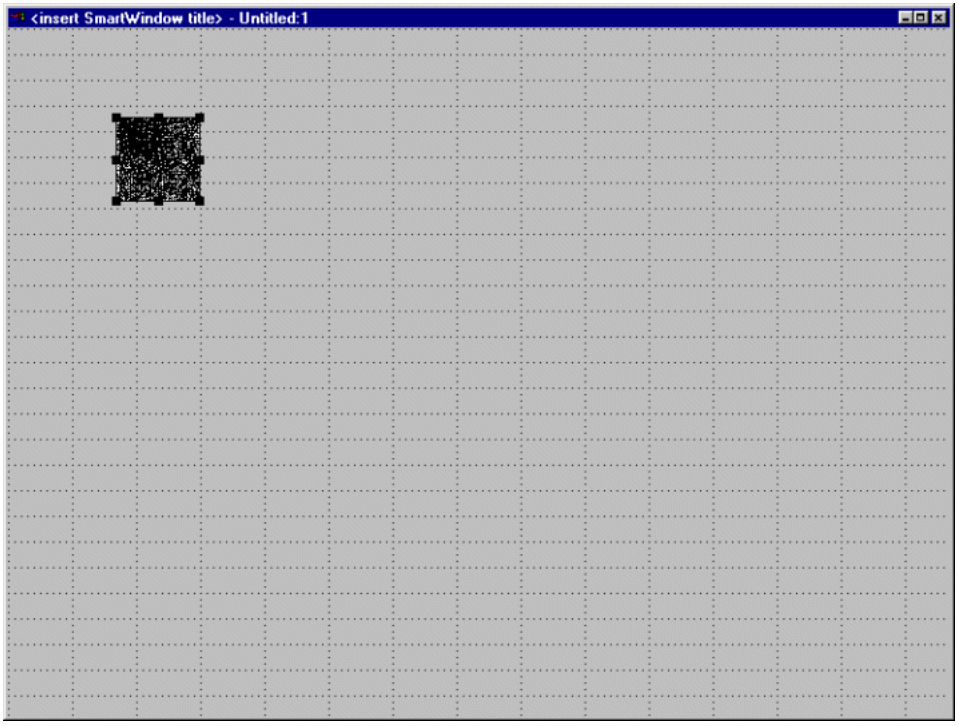
Defining Sections Of the Main Window

In this section, you add a graphic banner to the main window. After adding the banner, you define the Customer, Order, and Order Line sections.

Follow these steps to define the sections in the main window:

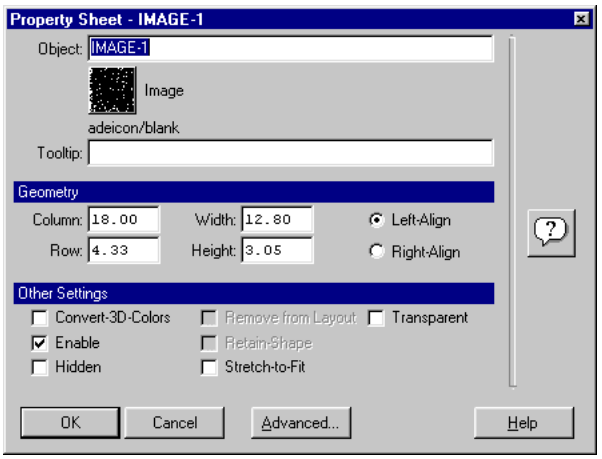
1 ♦ Choose the Image button  in the Object Palette and place one on your window.

A black square opens in the main window:



2 ♦ Double-click the square to open its property sheet.

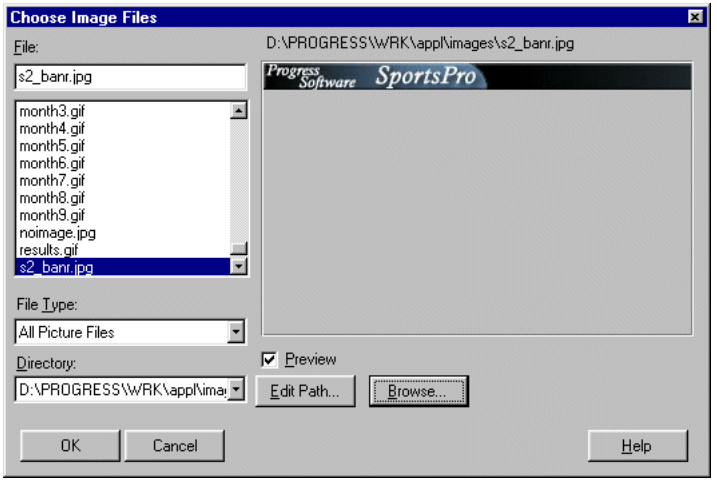
The IMAGE–1 property sheet opens:



3 ♦ Choose the Image button to open the Choose Image Files dialog. Browse to the images folder in your working directory. By default, this is the `\Progress\wrk\images` directory.

4 ♦ Select the `s2_banr.jpg` image and choose Open.

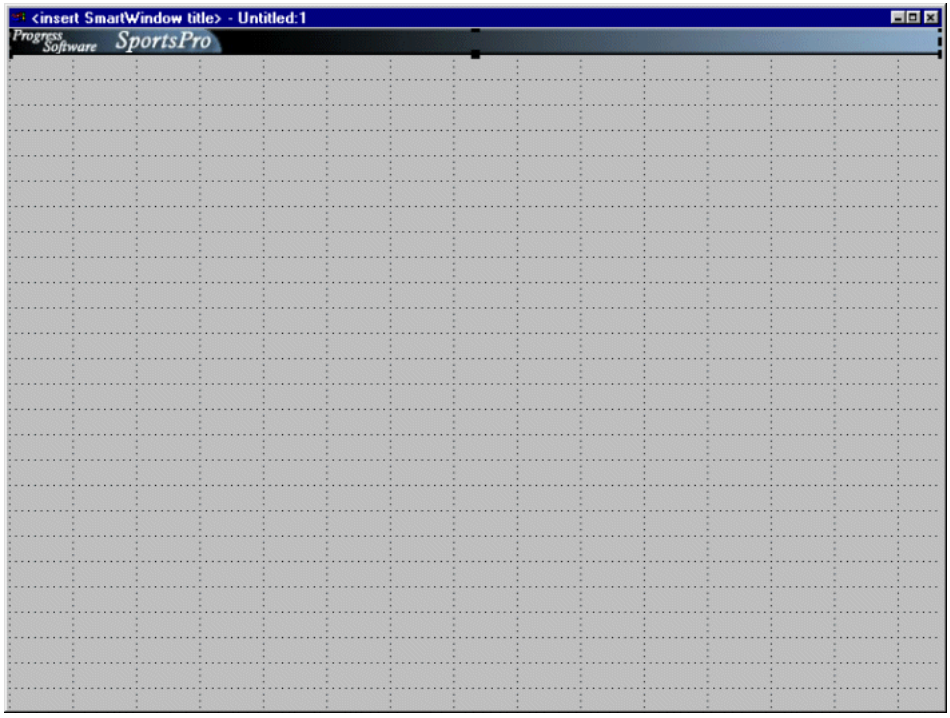
The `s2_banr.jpg` file and the other image files appear in the File list. The selected image previews to the right of the file list:



NOTE: The image appears in the Preview section only if you have selected the Preview option.

5 ♦ Accept the selected image and close the property sheet to return to your workspace.

6 ♦ Resize the banner and position it, as shown in the following figure:



Outlining the Customer, Order, and Order Line Sections

After creating the main window and adding the banner, your next task is to divide the window into Customer, Order, and Order Line sections.

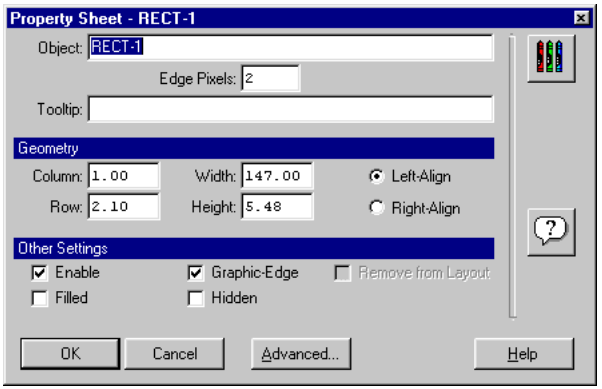
Follow these steps to outline the sections:

1 ♦ Double-click the Rectangle icon  in the Object Palette. A lock appears on the icon indicating that you can add multiple rectangles without reselecting the icon for each.

NOTE: To identify an object in the palette, select the Object Palette and pass the cursor over an object. As the pointer passes over an object, a ToolTip identifies that object. For more information about the Object Palette, see Chapter 1, "Overview Of the Progress Application Development Environment."

2 ♦ Draw rectangles to roughly outline the Customer, Order, and Order Line sections.

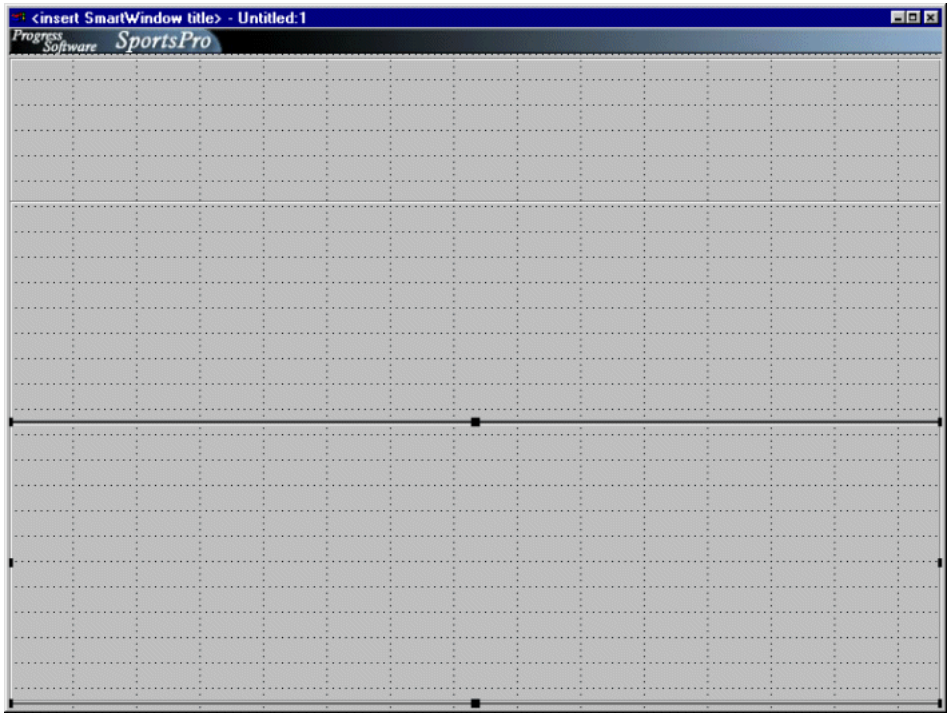
3 ♦ Select each rectangle in turn and open its property sheet:



4 ♦ Set the following values for each rectangle's Geometry attributes:

Rectangle	Column	Row	Width	Height
RECT-1	1.00	2.10	147	5.48
RECT-2	1.00	7.48	147	8.33
RECT-3	1.00	15.81	147	10.67

The AppBuilder resizes the rectangles to reflect the new settings. Your main window should now look like this:



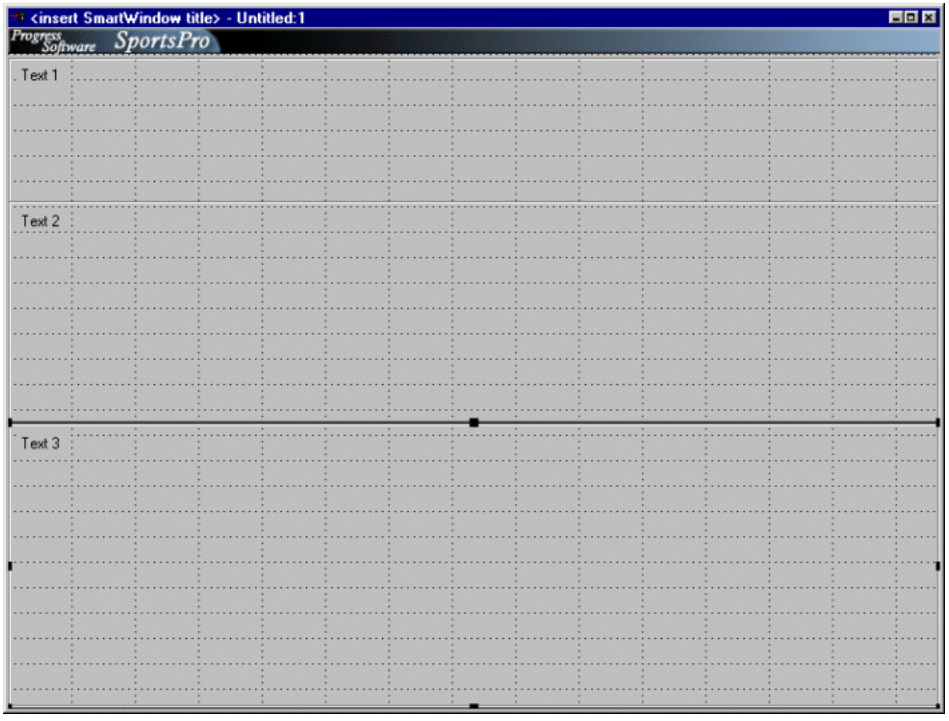
Adding Text Labels To the Sections

Now that you have divided your main window into sections for Customer, Order, and Order Line information, you need to add text labels to identify the sections.

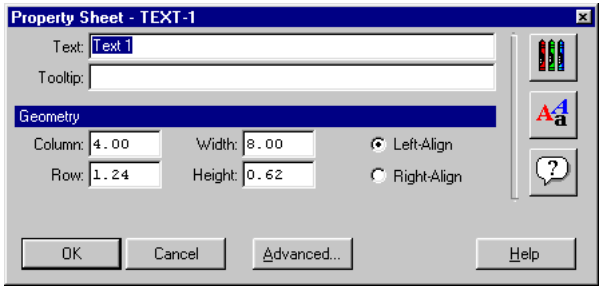
Follow these steps to add text labels:

1 ♦ Double-click the Text icon  in the Object Palette.

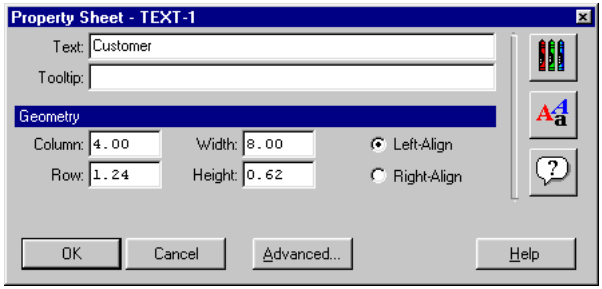
2 ♦ Place a text label in each section of the SmartWindow, as shown:



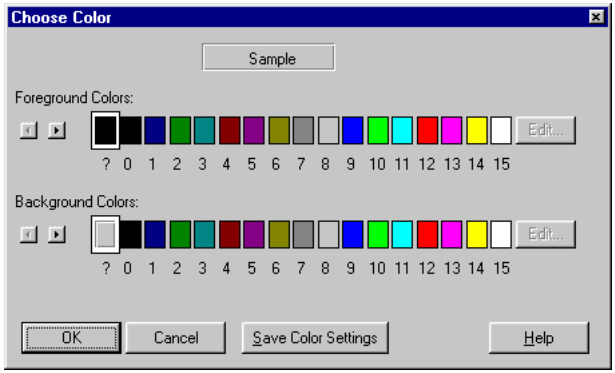
3 ♦ Open the property sheet for the text label in the Customer section:



4 ♦ In the Text field, type **Customer**:



5 ♦ Choose the Color icon  to open the Choose Color dialog box:



6 ♦ Under Foreground Colors, choose the dark blue color, setting 1.

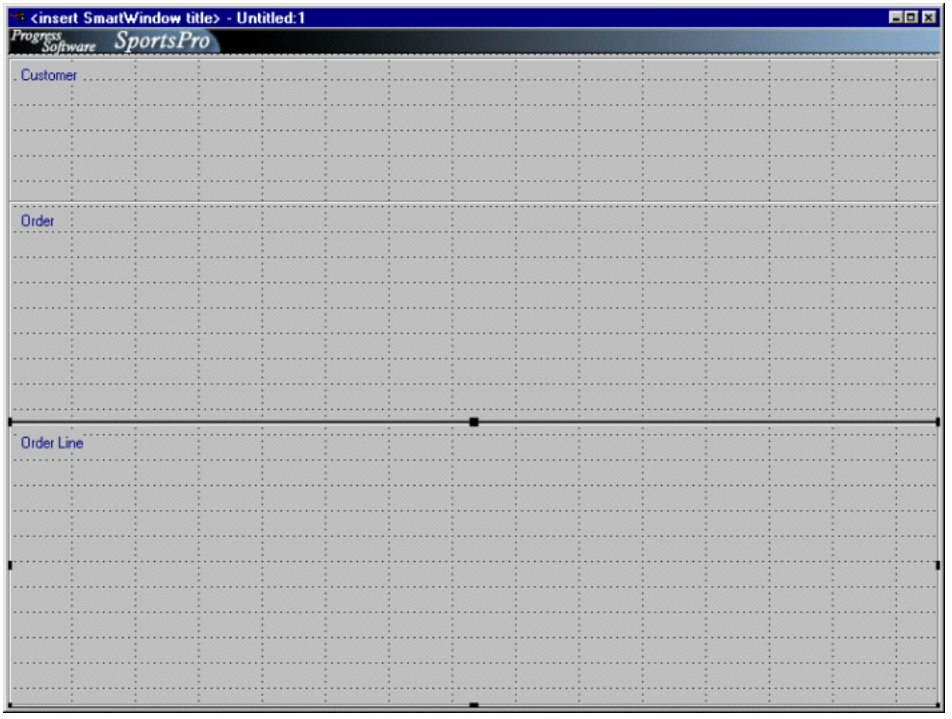
NOTE: The colors labeled with a question mark indicate the default colors.

7 ♦ Commit the changes and return to the main window.

8 ♦ Resize the text label so that all the text is visible.

9 ♦ Return to Step 3 and repeat the procedure for the Order and Order Line text labels.

When you are finished, the window looks like the following:

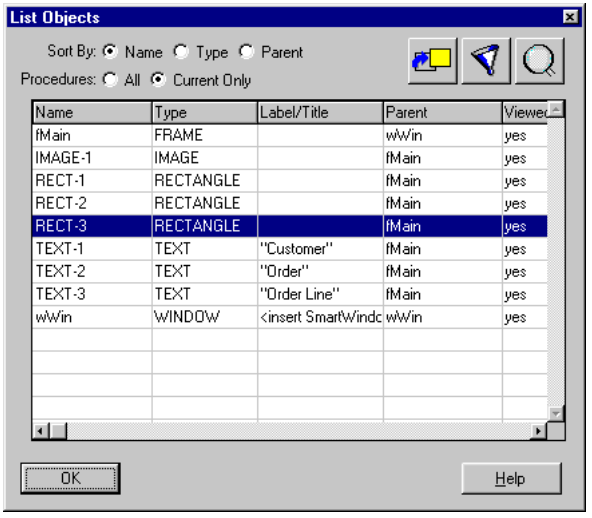


Naming and Saving the New SmartWindow

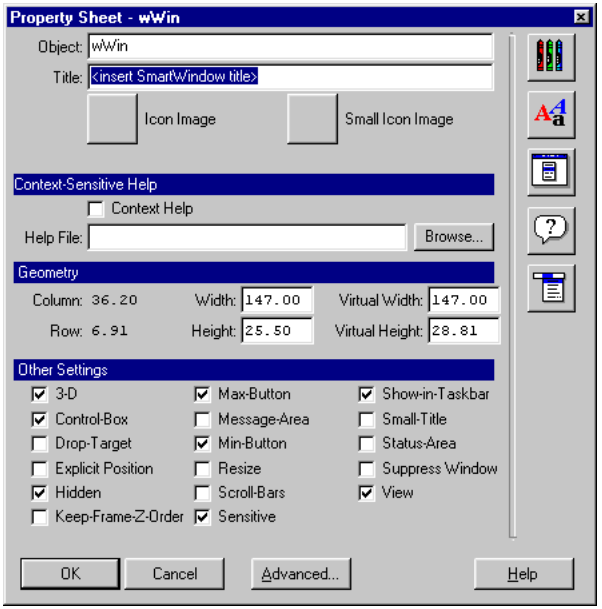
Now that you have your main window's basic design, you should name the SmartWindow and save your work before adding SmartObjects to it.

Follow these steps to name the window and save your work:

1 ♦ Choose the List Objects icon  on the AppBuilder toolbar to open the List Objects dialog box:



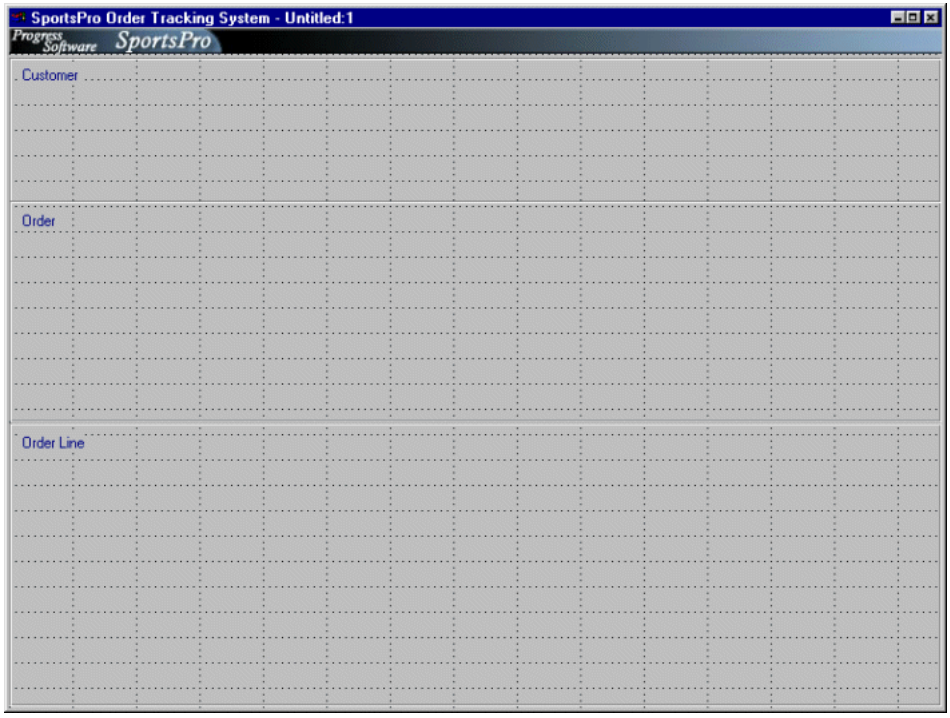
2 ♦ From the list of objects, double-click the WINDOW object to open the property sheet:



3 ♦ In the Title field, type **SportsPro Order Tracking System** and commit the change.

4 ♦ Return to your window.

The new window title appears in the title bar:

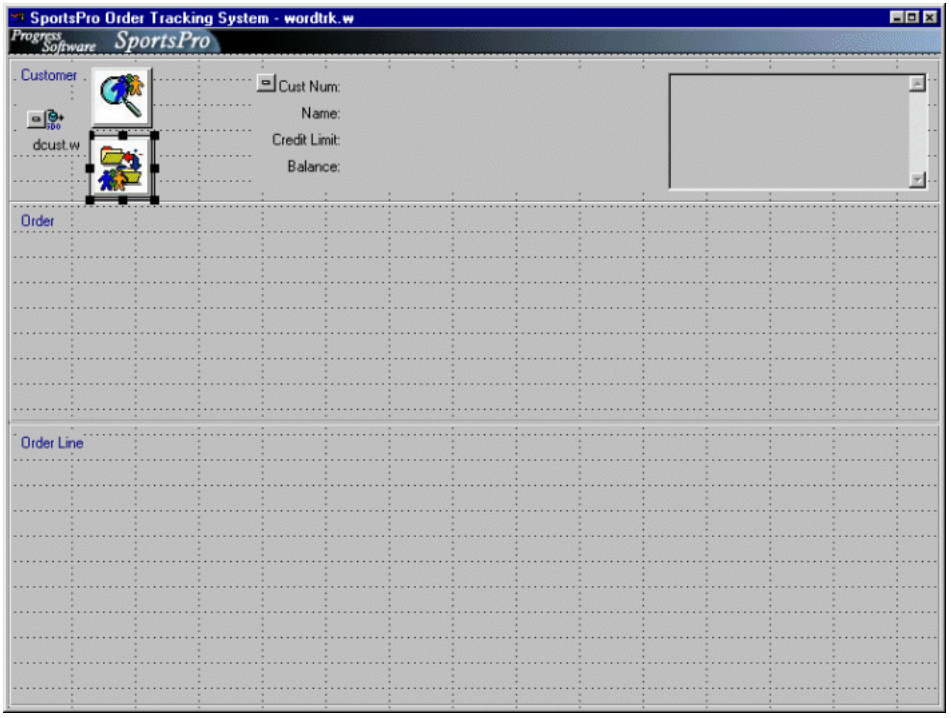


5 ♦ Save your file as **wordtrk.w** in your working directory.

You are now ready to add objects to your window.

3.4 Building the Customer Section

The Customer section of the SportsPro Order Tracking System contains a SmartDataObject (SDO) for querying customer information, a SmartDataViewer (SDV) for viewing customer information, and buttons to launch the Find Customer and Customer Maintenance modules. When you are finished with this section of the tutorial, your window should look something like this:



3.4.1 Task List and Short Procedure

To reach this point, you need to know how to do the following tasks:

1. Add a prebuilt SDO.
2. Add a prebuilt SDV.
3. Link an SDV to an SDO.
4. Create and resize a button.

5. Enhance a button with a ToolTip, a Help ID, and an image.

6. Link a Help file to a window.

7. Code a launch trigger for a button using the Section Editor.

The following short procedure outlines how to complete these tasks:

1 ♦ Add a prebuilt SDO, `dcust.w`, to the Customer section.

To review this procedure, see the "Adding a SmartDataObject" section.

2 ♦ Add a prebuilt SDV, `vcust.w`, to the Customer section.

To review this procedure, see the "Adding a SmartDataViewer" section.

3 ♦ Create a Data SmartLink from the SDO to the SDV.

To review this procedure, see Step 4 in the "Adding a SmartDataViewer" section.

4 ♦ Create an Update SmartLink from the SDV to the SDO.

To review this procedure, see Step 5 in the "Adding a SmartDataViewer" section.

5 ♦ Create two buttons with the following properties:

Object	Column	Row	Height	Width	ToolTip	Help ID	Image
BUTTON-1	14	2.42	2.38	10	Find Customer	1	`Findcust.jpg`
BUTTON-2	14	5.04	2.38	10	Maintain Customer	2	`mntcust.jpg`

NOTE: The image files are stored in the `images` subdirectory of your working directory.

To review this procedure, see the "Creating and Positioning the Buttons" and "Adding ToolTips, Help IDs, and Images To the Buttons" sections.

6 ♦ Link a Help file to your window.

To review this procedure, see the "Adding Help Text To the Window" section.

7 ♦ Use the Section Editor to write CHOOSE event triggers for the buttons.

To review this procedure, see the "Coding Event Triggers For the Buttons" section.

3.4.2 Detailed Customer Section Procedure

The following sections lead you through assembling the Customer section in more detail.

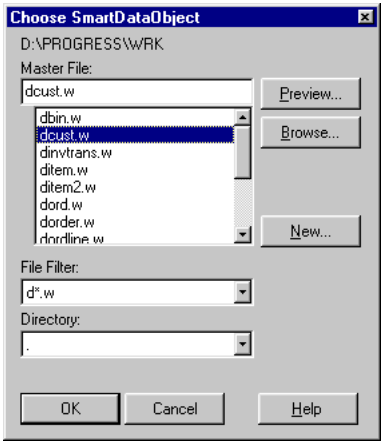
Adding a SmartDataObject

The first SmartObject added in most applications is the SDO that acts as a data source for the rest of the application's SmartObjects.

Follow these steps to add an SDO:

1 ♦ Choose the SDO icon 

in the Object Palette to open the Choose SmartDataObject dialog box:



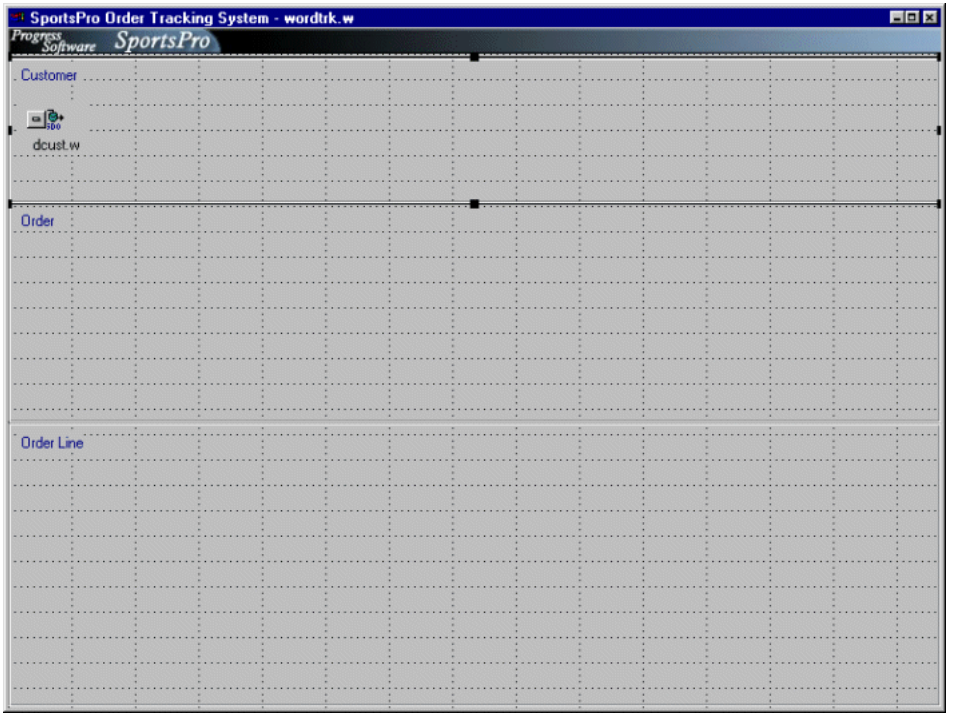
2 ♦ Select `dcust.w` and choose OK.

3 ♦ Move the object to the Customer section.

As your pointer passes over the main window, the pointer resembles the SDO icon.

4 ♦ Click once in the Customer section. You do not need to establish an exact position for an SDO because it is invisible when the application is running.

The new SDO appears in the Customer section of the window:

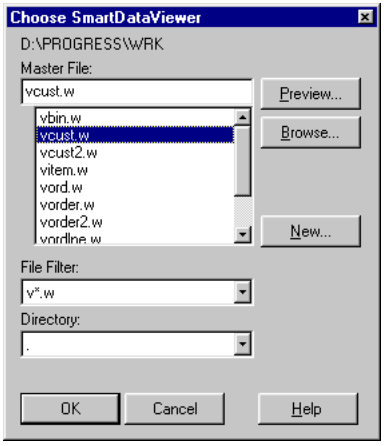


Adding a SmartDataViewer

The next object to add is an SDV. The SDV displays the customer information retrieved by the SDO.

Follow these steps to add an SDV:

1 ♦ Choose the SDV icon  in the Object Palette to open the Choose SmartDataViewer dialog box:

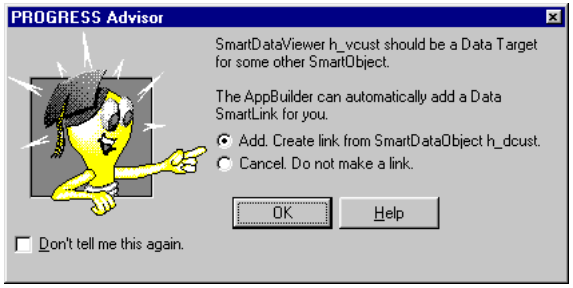


2 ♦ Select `vcust.w` and choose OK.

3 ♦ Drag the object to the Customer section. As you pass the pointer over the main window, the pointer resembles the SDV icon.

4 ♦ Click once in the Customer section.

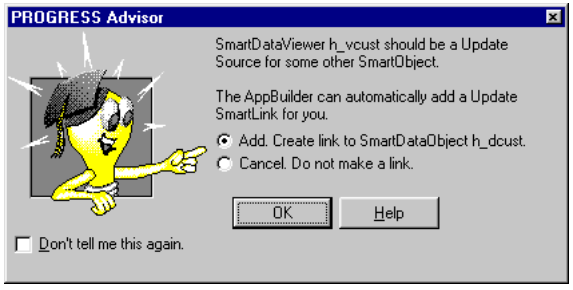
The Progress Advisor opens, notifies you that the SDV instance, `h_vcust`, should be a data target for another SmartObject, and offers to create a Data SmartLink from the SDO instance, `h_dcust`:



NOTE: The Progress Advisor refers to SmartObjects with an `h_ prefix`, for example, `vcust` as `h_vcust`, to indicate a handle. A handle refers to an instance of a SmartObject in memory.

5 ♦ Select the radio button to create the SmartLink and choose OK.

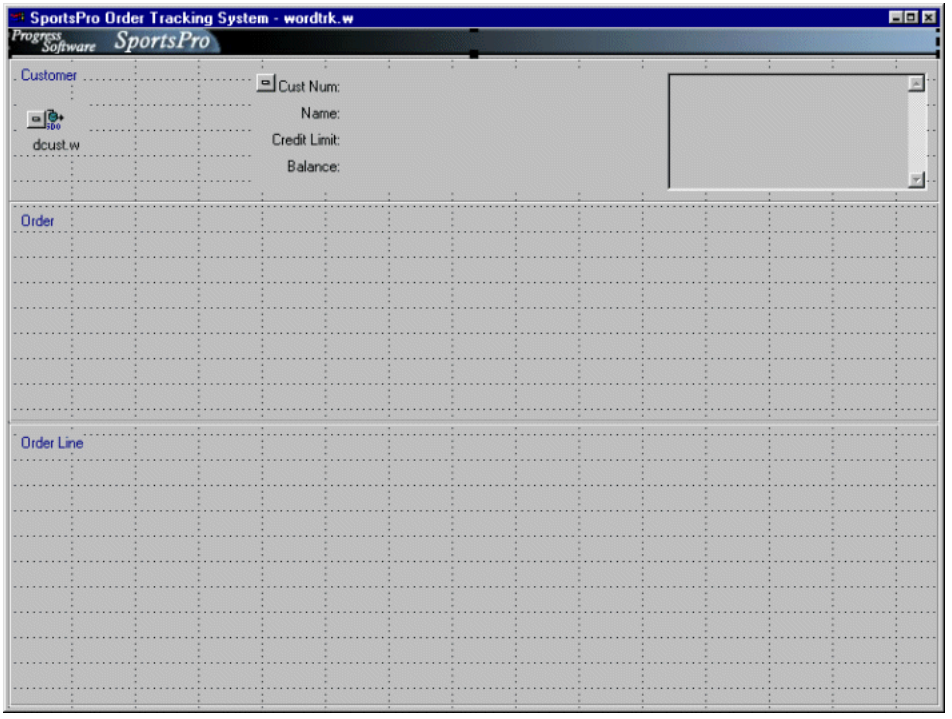
The Progress Advisor opens again, notifies you that the SDV instance, `h_vcust`, should be an Update Source for some other SmartObject, and offers to create an Update SmartLink to the SDO instance, `h_dcust`:



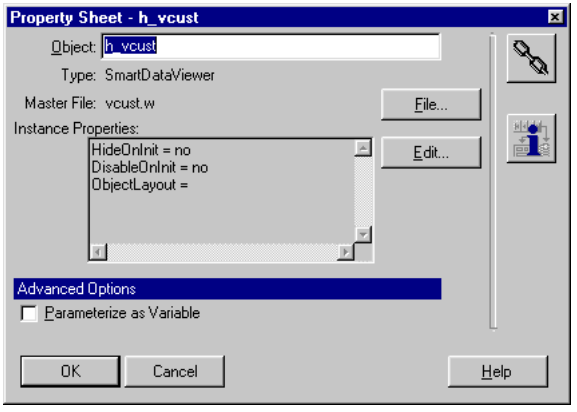
6 ♦ Select the radio button to create the SmartLink and choose OK.

The SDV, `vcust`, appears in the Customer section.

7 ♦ Position the SDV within the Customer section, aligning it on the right edge of the design window, as shown in the following figure:

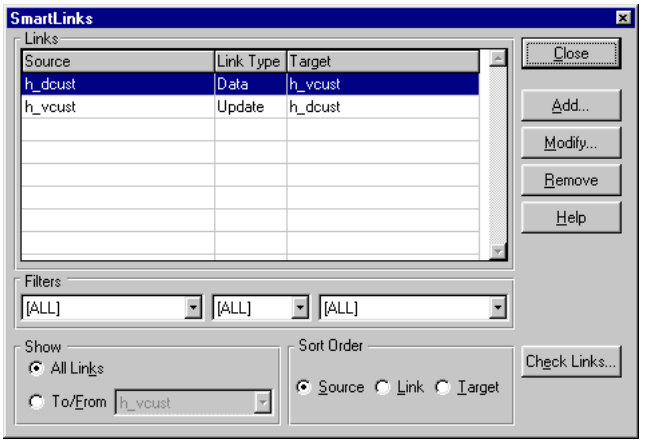


8 ♦ Double-click the SDV to open its property sheet:



9 ♦ Choose the SmartLinks button  to view SmartLink information:

The SmartLinks dialog box opens showing a Data SmartLink from `h_dcust` to `h_vcust` and an Update SmartLink from `h_vcust` to `h_dcust`:



10 ♦ Close the SmartLinks dialog box and property sheet.

11 ♦ Save your work.

Your next task is to add two buttons to the Customer section, complete with ToolTips, images, and help text.

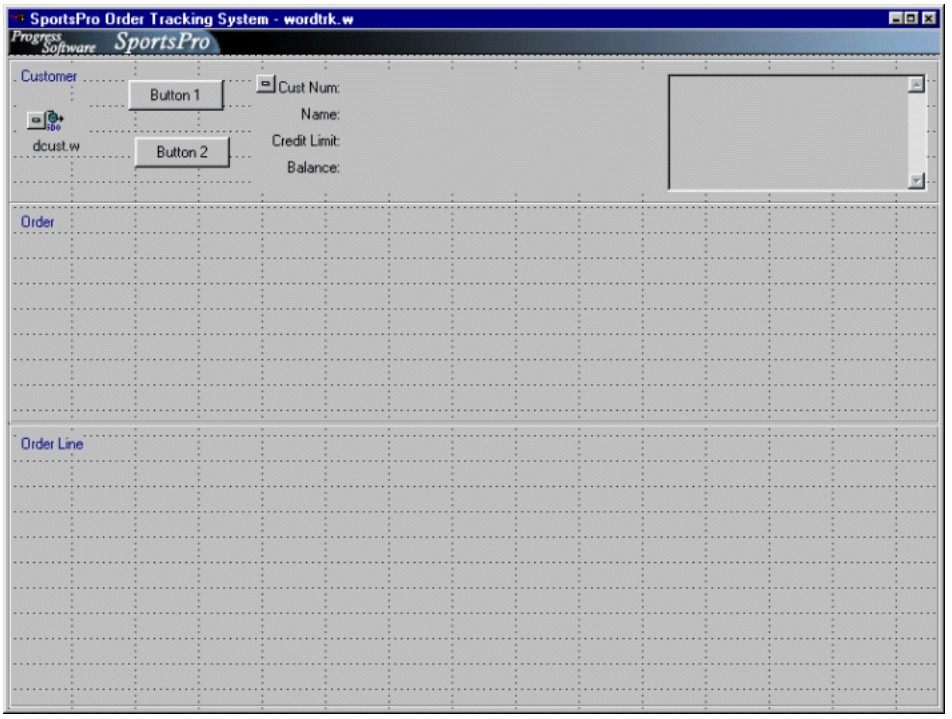
Creating and Positioning the Buttons

The first step in adding buttons to your application is to create and position them.

Follow these steps to create and position the buttons:

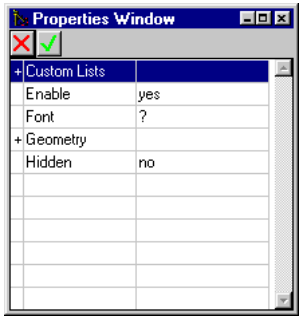
1 ♦ Double-click the Button icon  in the Object Palette.

2 ♦ Click in the Customer section in two places to add two buttons, as shown. Do not position the buttons exactly. You will use a Properties Window to specify precise settings later. The main window should look something like the following figure:

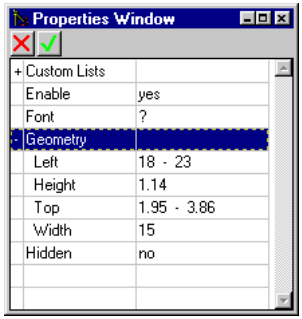


3 ♦ Choose the Pointer icon and, holding down the **CTRL** key, select both buttons. Then choose Window→ Properties Window from the AppBuilder menu.

The Properties Window is a means to quickly set common attributes in many objects at once:



4 ♦ Double-click to expand the options under the Geometry row:

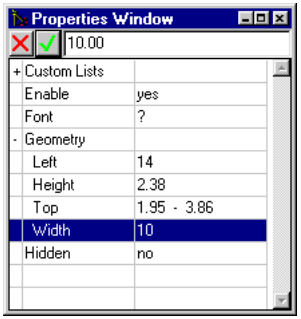


NOTE: In this Properties Window, you can only change one value at a time. Select a row, type a value in the fill-in field next to the check-mark button, and then choose the check-mark to commit the change.

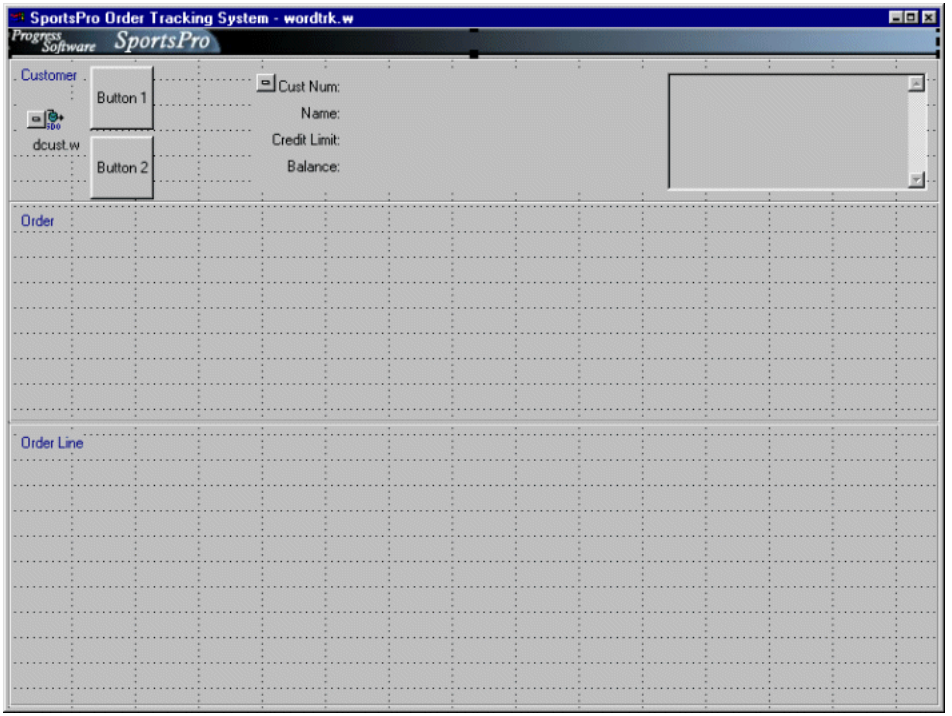
5 ♦ Set the following attributes for the buttons:

Attribute	Value
Column	14
Height	2.38
Width	10

When you finish, the property sheet should looks like this:



6 ♦ Close the Properties Window and manually adjust the buttons' vertical positions to resemble the following figure:



Adding ToolTips, Help IDs, and Images To the Buttons

After creating and positioning the buttons, you are ready to add a ToolTip, a Help ID, and an image to each button.

Follow these steps to add ToolTips, images, and help IDs to the buttons:

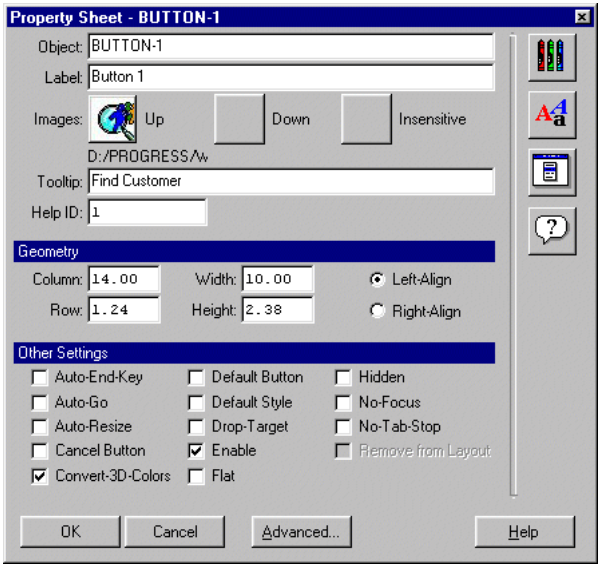
1 ♦ Open the property sheet for BUTTON-1 and set the following properties:

Property	Value
Tooltip	Find Customer
Help ID	1

2 ♦ Choose the Up button in the Images section to open the Choose Image dialog box.

3 ♦ Select `Findcust.jpg` in the `images` folder in your working directory.

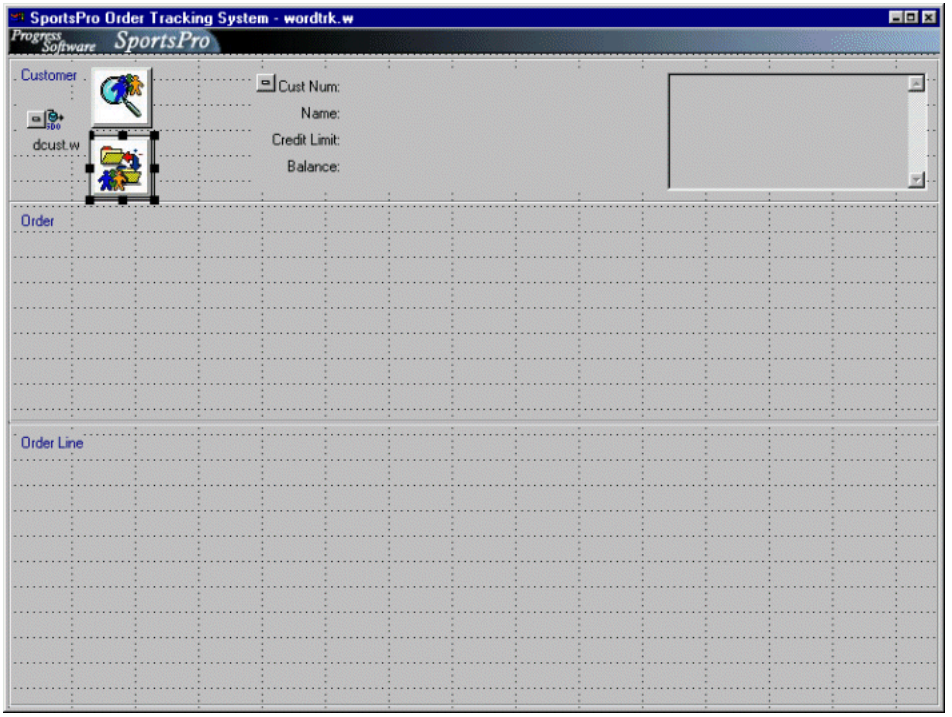
The button's property sheet now looks like this:



4 ♦ Repeat the procedure for BUTTON-2, setting the following properties:

Property	Value
Tooltip	Maintain Customer
Help ID	2
Up Image	`mntcust.jpg`

When you finish, your SmartWindow should appear as follows:

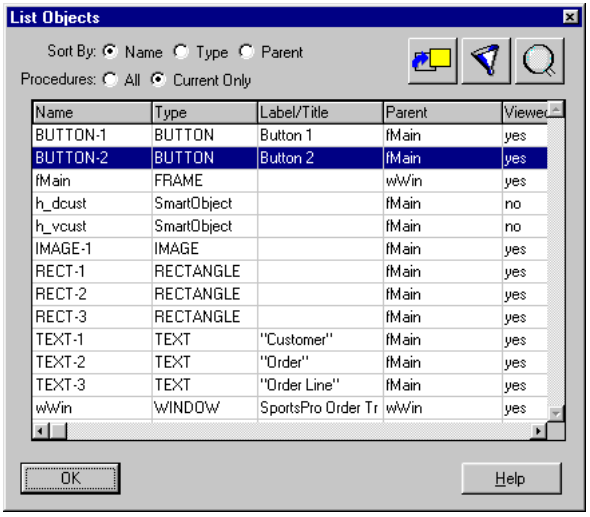


Adding Help Text To the Window

Now that you have added Help IDs, you need to link a context-sensitive help file to your application. You add a help file for a window. All of the simple objects on that window reference the same context-sensitive help.

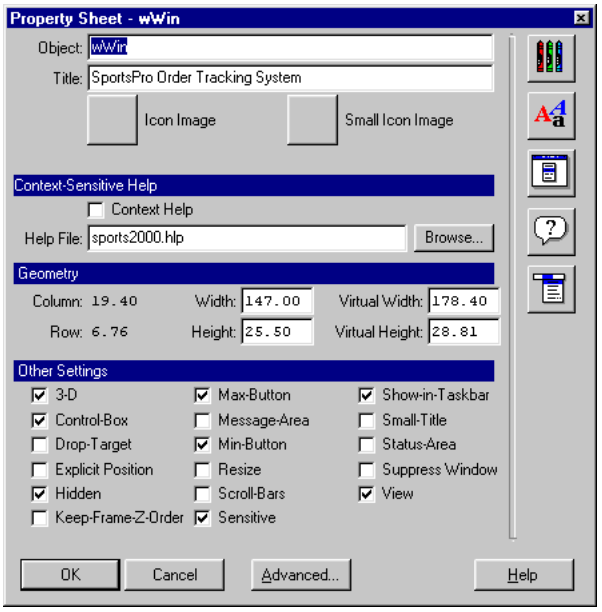
Follow these steps to add a context-sensitive help file:

1 ♦ Choose the List Object icon  on the AppBuilder toolbar to open the List Objects dialog box:



2 ♦ Select wWin from the Name list to select the window object.

3 ♦ Choose the Object Properties icon  in the AppBuilder toolbar to open the property sheet for the window:



4 ♦ Choose the Context Help toggle-box to activate context sensitive help.

5 ♦ Type **sports2000.hlp** in the Help File text box and choose OK to close the dialog box.

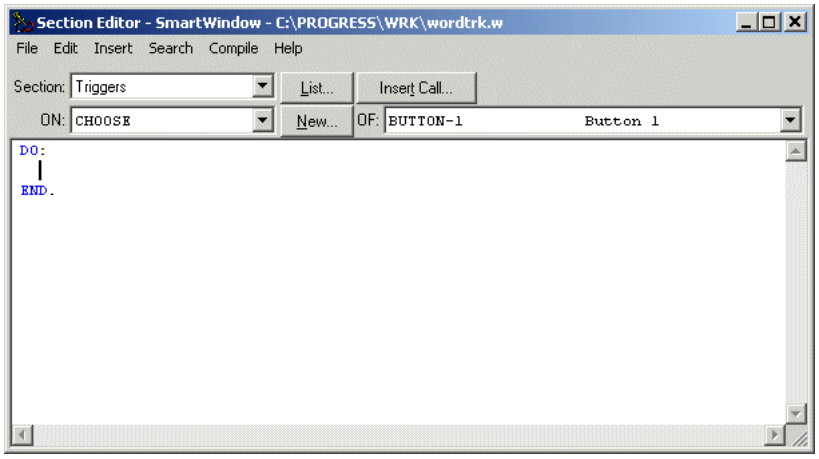
6 ♦ Save your work.

Coding Event Triggers For the Buttons

Your next step is to add code to the buttons. When the user clicks on either button, a CHOOSE event occurs. Using the Section Editor, you can quickly add code to launch the appropriate module when each CHOOSE event occurs.

To launch the appropriate module when the user chooses a button, follow these steps to add the trigger code:

1 ♦ Select BUTTON-1 and open the Section Editor with the Edit Code icon  in the AppBuilder toolbar:

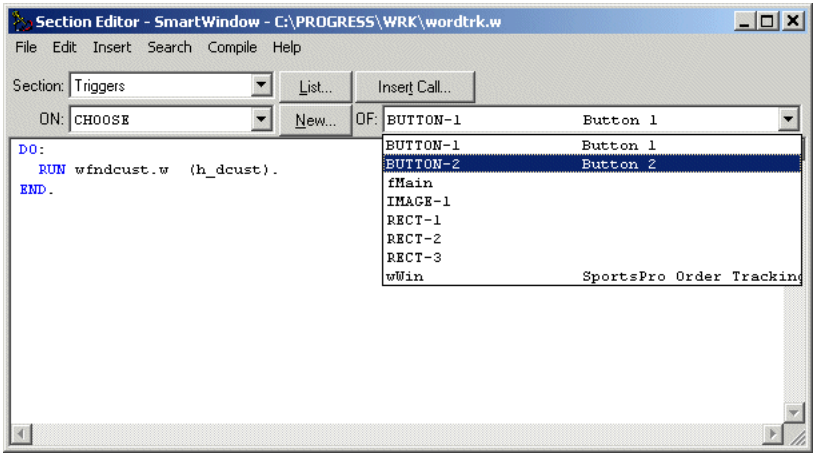


2 ♦ Between the DO and END statements, add the following code:

```
RUN wfndcust.w  (h_dcust).
```

NOTE: This statement passes the handle for the `dcust` SDO to the module you are starting. As a result, the new module opens on the same customer record. Because the new module does not have an internal data source, it cannot run on its own.

3 ♦ Select BUTTON-2 in the Object combo-box.



4 ♦ Between the DO and END statements, add the following code:

```
RUN wmntcust.w (h_dcust).
```

5 ♦ Choose File→ Close Window or the Close Window button to save the new code and return to the workspace.

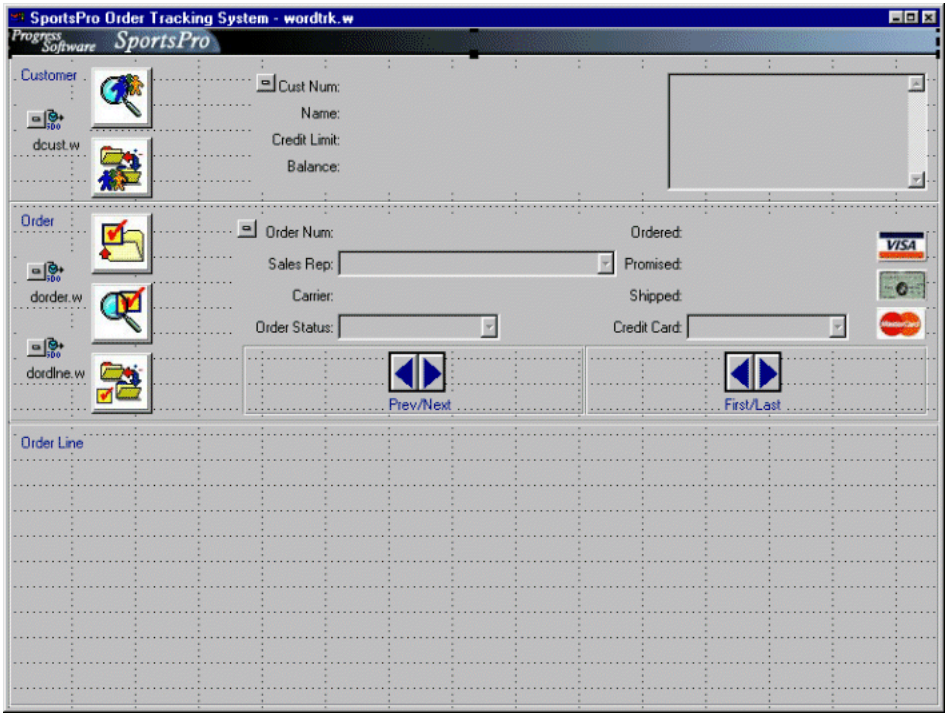
NOTE: If you do not want to save code that you entered in the Section Editor, you must use the Edit→ Undo menu options.

6 ♦ Save your work.

Now that you have built the Customer section of the SmartWindow, you can add SmartObjects to the Order section.

3.5 Building the Order Section

The Order section contains SmartObjects for querying and viewing order information: SDOs, an SDV, Object Component Extension (OCX) controls (also known as ActiveX controls), and buttons for launching associated modules. When you are finished with this section of the tutorial, your window should look something like this:



3.5.1 Task List and Short Procedure

To reach this point, you need to know how to do the following tasks:

1. Add a prebuilt SDO.
2. Link SDOs together and map their foreign fields.
3. Add a prebuilt SDV.
4. Link an SDV to an SDO.
5. Create and resize a button.
6. Enhance a button with a ToolTip, a Help ID, and an image.
7. Code a launch trigger for a button using the Section Editor.
8. Create and resize an OCX control.
9. Code navigation triggers for an OCX control using the Section Editor.

The following short procedure outlines how to complete these tasks:

1 ♦ Add a prebuilt SDO, `dorder.w`, to the Order section.

To review this procedure, see the "Adding a SmartDataObject For Orders" section.

2 ♦ Create a Data SmartLink from `dcust.w` to `dorder.w`.

To review this procedure, see Step 3 in the "Adding a SmartDataObject For Orders" section.

3 ♦ Map the `Customer.CustNum` field to the `Order.CustNum` field.

To review this procedure, see Step 4 in the "Adding a SmartDataObject For Orders" section.

4 ♦ Add a prebuilt SDO, `dordlne.w`, to the Order section.

To review this procedure, see the "Adding a SmartDataObject For Order Lines" section.

5 ♦ Create a Data SmartLink from `dorder.w` to `dordlne.w`.

To review this procedure, see Step 3 in the "Adding a SmartDataObject For Order Lines" section.

6 ♦ Map the `Order.Ordernum` field to the `OrderLine.Ordernum` field.

To review this procedure, see Step 5 in the "Adding a SmartDataObject For Order Lines" section.

7 ♦ Add a prebuilt SDV, `vorder.w`, to the Order Section.

To review this procedure, see the "Adding a SmartDataViewer" section.

8 ♦ Create a Data SmartLink from `dorder.w` to `vorder.w`.

To review this procedure, see Step 3 in the "Adding a SmartDataViewer" section.

9 ♦ Create an Update SmartLink from `vorder.w` to `dorder.w`.

To review this procedure, see Step 4 in the "Adding a SmartDataViewer" section.

10 ♦ Add three buttons with the following properties:

Object	Column	Row	Height	Width	ToolTip	Help ID	Image
BUTTON-3	14	7.92	2.38	10	Place New Order	3	`crtord.jpg`
BUTTON-4	14	10.54	2.38	10	Find Order	4	`findord.jpg`
BUTTON-5	14	13.15	2.38	10	Maintain Order	5	`mainord.jpg`

To review this procedure, see "Creating and Positioning the Buttons" and "Adding ToolTips, Help IDs, and Images To the Buttons" in the "Detailed Customer Section Procedure" section.

11 ♦ Use the Section Editor to write CHOOSE event triggers for the buttons.

To review this procedure, see the "Coding Event Triggers For the Order Buttons" section.

12 ♦ Add and position two control frames, based on the Crescent Spin Control OCX.

To review this procedure, see the "Adding Object Component Extensions" section.

13 ♦ Give the OCX control frames the following properties:

Property	Value
Height	1.52
Width	9.0
Style	3D Horizontal
ArrowColor	dark blue

To review this procedure, see the "Changing the Appearance Of the OCX Controls" section.

14 ♦ Create the following Text labels:

Object	Label	Color
CtrlFrame	Prev/Next	1
CtrlFrame-2	First/Last	1

To review this procedure, see Step 7 in the "Changing the Appearance Of the OCX Controls" section.

15 ♦ Place a rectangle around each OCX control and its text label.

To review this procedure, see Step 9 in the "Changing the Appearance Of the OCX Controls" section.

16 ♦ Use the Section Editor to add OCX.SpinDown and OCX.SpinUp triggers for each control frames.

To review this procedure, see the "Coding the Prev/Next Spin Box To Fetch the Previous Record" and "Coding the First/Last Spin Box To Fetch the First Record" sections.

17 ♦ Run and check your application.

To review this procedure, see the "Viewing the Window" section.

3.5.2 Detailed Order Section Procedure

The following sections describe building the Order section in detail.

Adding a SmartDataObject For Orders

As in the Customer section, the SDOs are the first objects to add to the Order section. The Order section needs SDOs that query the database for orders and the order lines that make up that order.

Follow these steps to add an SDO:

1 ♦ Choose the SDO icon 

in the Object Palette to open the Choose SmartDataObject dialog box.

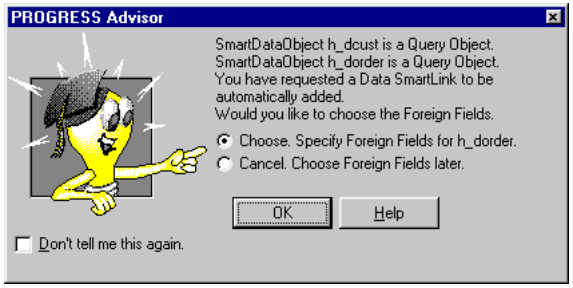
2 ♦ Select `dorder.w` and choose OK.

3 ♦ Click once in the Order section. You do not need to position a SmartDataObject precisely, since it is invisible when an application is running.

The Progress Advisor opens, notifies you that the SDO instance, `h_dorder`, can be a Data Target for some other SmartObject, and offers to add a Data SmartLink from the SDO instance, `h_dcust`.

4 ♦ Select the radio button to create the SmartLink and choose OK.

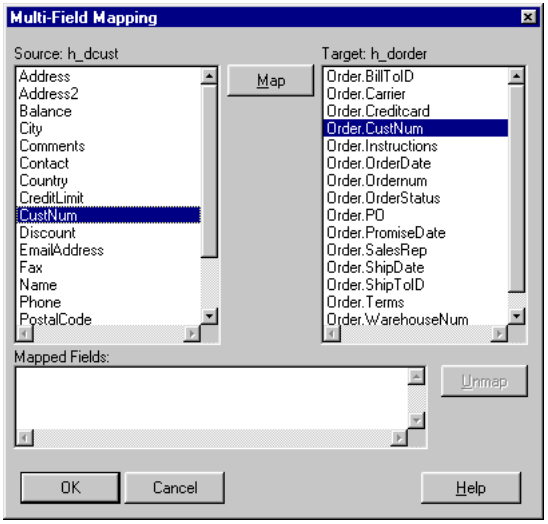
The Progress Advisor opens, notifies you that `h_dorder` and `h_dcust` are both Query Objects, and asks whether you want to specify foreign fields:



5 ♦ Select the radio button to choose foreign fields and choose OK.

The Multi-Field Mapping dialog box opens.

6 ♦ Map `CustNum` in the Source list to `Order.CustNum` in the Target list, as shown in the following figure:



The new SmartDataObject, `dorder.w`, appears in the main window when you choose OK.

Adding a SmartDataObject For Order Lines

You are now going to add another SDO, `dordlne`, to the Order section.

Follow these steps to add the SDO:

1 ♦ Choose the SDO icon 

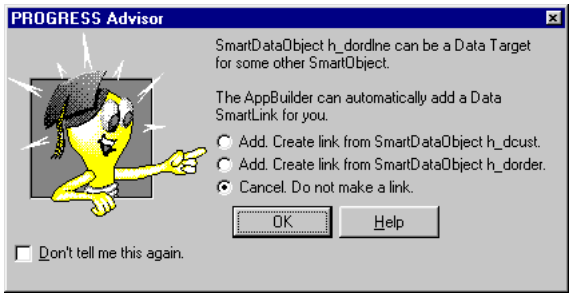
in the Object Palette to open the Choose SmartDataObject dialog box.

2 ♦ Select `dordlne.w` and choose OK.

3 ♦ Position the pointer below the other SDO, `dorder`, and click once.

NOTE: Because the SDO is invisible when the application runs, you do not need to position the SDO precisely.

The Progress Advisor opens, notifies you that the SDO instance, `h_dordlne`, can be a Data Target for some other SmartObject, and offers to add a Data SmartLink from either of the other SDO instances:



4 ♦ Choose the radio button to create the SmartLink and choose OK.

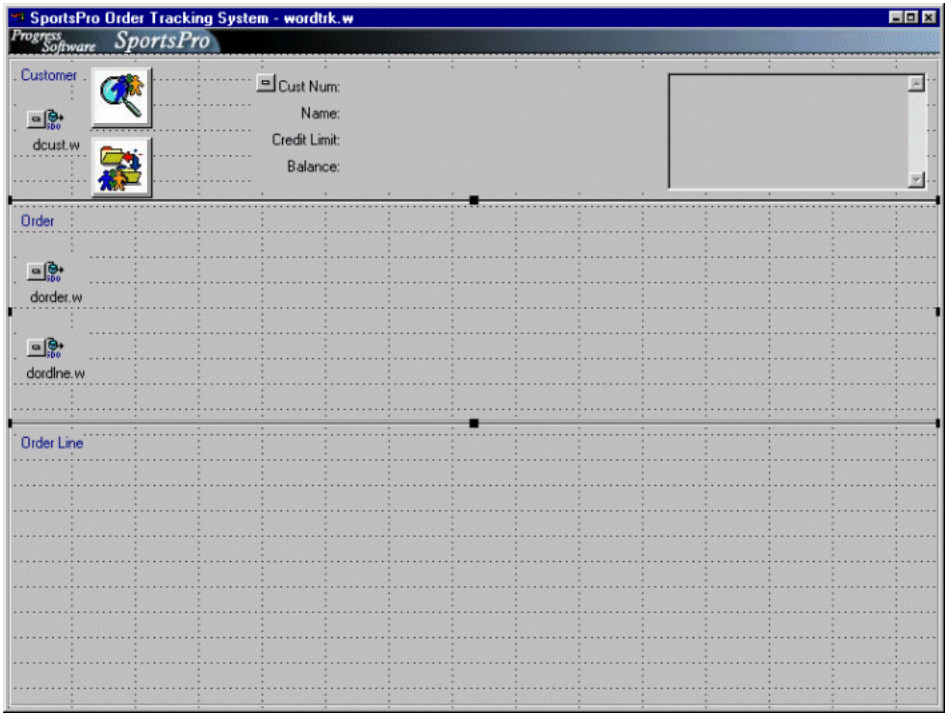
The Progress Advisor opens, notifies you that `h_dordlne` and `h_dorder` are both Query Objects and asks if you want to choose foreign fields.

5 ♦ Choose the radio button to select foreign fields and then choose OK.

The Multi-Field Mapping dialog box opens.

6 ♦ Map `Ordernum` in the Source list to `OrderLine.Ordernum` in the Target list.

The new SmartDataObject, `dordlne`, appears in the design window when you choose OK:



7 ♦ Save your work.

Adding a SmartDataViewer

The next object to add is an SDV for displaying order information.

Follow these steps to add an SDV:

1 ♦ Choose the SDV icon  in the Object Palette to open the Choose SmartDataViewer dialog box.

2 ♦ Select `vorder.w` and choose OK.

As you pass the pointer over the design window, the pointer resembles the SDV icon.

3 ♦ Click within the Order section to add the SDV.

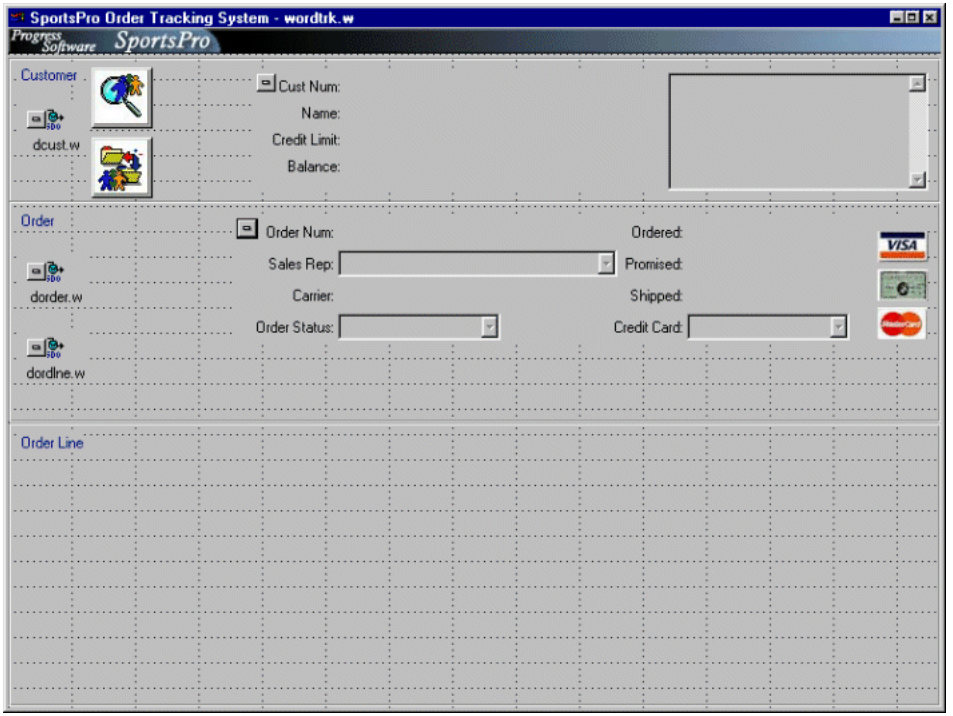
The Progress Advisor opens, notifies you that SDV instance, `h_vorder`, should be a Data Target for some other SmartObject, and offers to create a Data SmartLink from the SDO instance, `h_dorder`.

4 ♦ Select the radio button to create the SmartLink and choose OK.

The Progress Advisor opens, notifies you that the SDV instance, `h_vorder`, should be an Update source for some other SmartObject, and offers to add an Update SmartLink to the SDO instance, `h_dorder`.

5 ♦ Select the radio button to create the SmartLink and choose OK.

6 ♦ Position the SmartDataViewer near the top of the Order section and align it on the right edge of the Order section. Your SmartWindow should look like the one in the following figure:



Adding Buttons To the Order Section

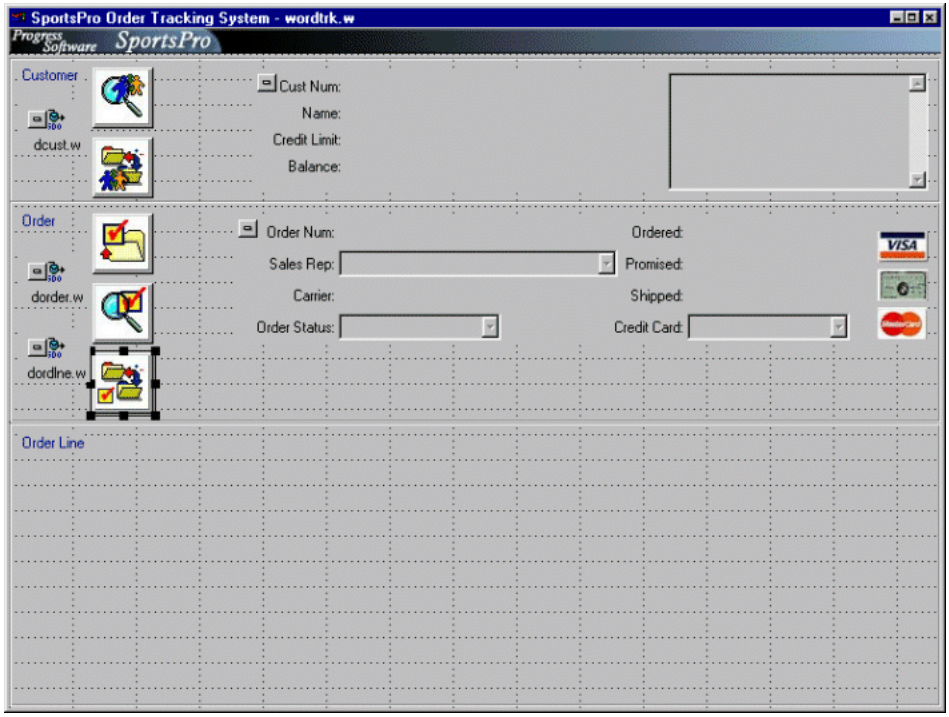
You need to add buttons to the Order section, as you did to the Customer section. Follow the procedures in "Creating and Positioning the Buttons" and "Adding ToolTips, Help IDs, and Images To the Buttons" in the "Detailed Customer Section Procedure" section of this chapter to add and customize the three buttons to the Order section.

Use the values in the following table to customize the buttons:

Object	Column	Row	Height	Width	ToolTip	Help ID	Image
BUTTON-3	14	7.92	2.38	10	Place New Order	3	`crtord.jpg`
BUTTON-4	14	10.54	2.38	10	Find Order	4	`findord.jpg`
BUTTON-5	14	13.15	2.38	10	Maintain Order	5	`mainord.jpg`

NOTE: You do not need to add a Help file for the Order section buttons. The Help file you added while building the Customer section controls the Help for the entire window. It applies to all sections of the window.

When you finish, your window should look like the following window:



Coding Event Triggers For the Order Buttons

You need to add CHOOSE event triggers to the buttons, as you did in the Customer section. Follow the procedures in "Coding Event Triggers For the Buttons" in the "Detailed Customer Section Procedure" section of this chapter.

Using the Section Editor, add the following code to the CHOOSE events of the buttons in this section:

Button	Code
BUTTON-3	`RUN wmntord.w (h_dcust, h_dorder, h_dordlne).`
BUTTON-4	`RUN wfndord2.w(h_dorder).`
BUTTON-5	`RUN wmntord.w (h_dcust, h_dorder, h_dordlne).`

The remaining objects to add to the Order section are two Object Component Extension (OCX) controls, also known as ActiveX controls.

Adding Object Component Extensions

In this section, you add Object Component Extension (OCX) controls to the SportsPro Order Tracking System. An OCX (also called an ActiveX control) is a predefined object widget that you include in a control-frame to help develop your applications.

ActiveX objects are Common Object Model (COM) objects that provide a form of interprocess communication with other Windows applications along the Microsoft ActiveX specifications. For more information about ActiveX Automation, see the *Progress External Program Interfaces* manual.

Progress ships with three licensed OCX controls:

- Crescent Spin Control, `csspin32.ocx`, the spin box
- CS ComboBox Control, `cscomb32.ocx`, a combo box
- Progress Timer Control, `pstimer.ocx`, an object that allows you to determine the length of time used by a process

You can select these OCX controls by:

- Selecting the OCX icon in the Object Palette and selecting the control from the Choose Control dialog box
- Selecting the specific OCX icon from the Object Palette

For this exercise, use the OCX icon to add the controls.

The AppBuilder enables you to integrate third-party objects, such as OCXs, into your applications similar to the way you build SmartObjects. For example, for the OCX spin box you work with in this section, you choose it from the Object Palette, position and resize it visually in the SmartDataViewer window, and write user-interface triggers for it using the Code Section Editor. For more detailed information about ActiveX (OCX controls) and the AppBuilder, see the *Progress AppBuilder Developer's Guide*.

In this section, you:

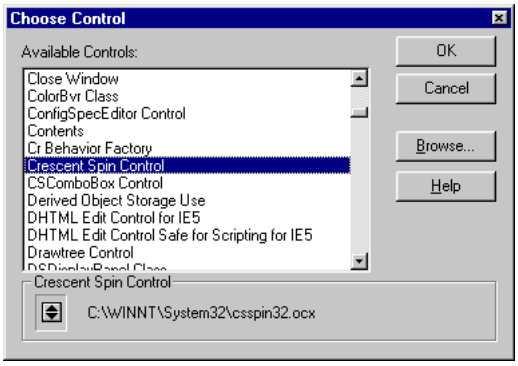
- Select OCX controls
- Change the appearance of the OCX controls
- Add user-interface triggers

Selecting an OCX Object

In this section, you add two OCX spin box controls to the Order section. These OCX controls are used to navigate through the records in the SDO, `dorder`, which sends the order information to the SDV, `vorder`, for viewing.

Follow these steps to add OCX controls to the Order section:

1 ♦ Choose the OCX icon  in the Object Palette to open the Choose Control dialog box:



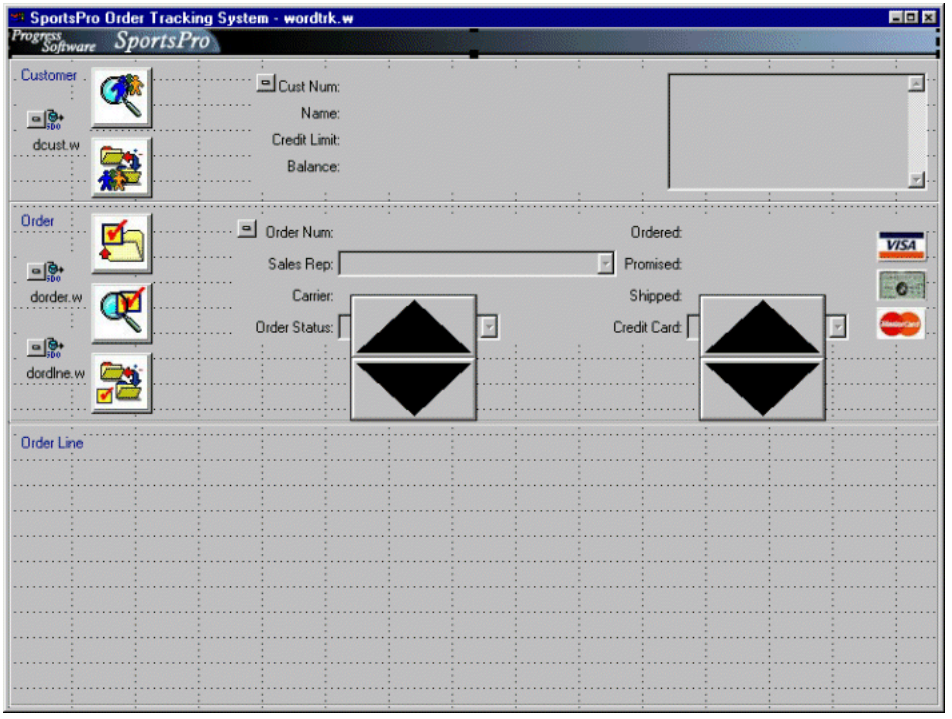
NOTE: If you installed other OCX controls, they should also appear in the Available Controls list.

2 ♦ Select the Crescent Spin Control and choose OK.

When you select an OCX control, the location and control name appear at the bottom of the Choose Control dialog box.

3 ♦ Add two spin boxes to the Order section.

The spin boxes appear with their default dimensions. The Order section should appear similar to the following window:



Changing the Appearance Of the OCX Controls

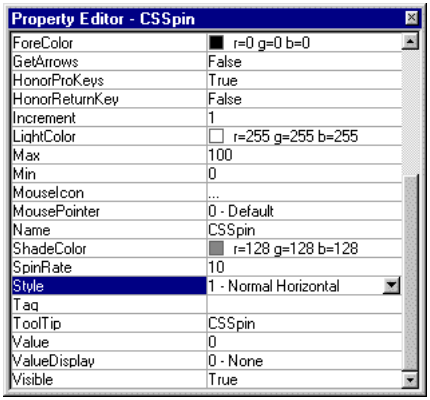
The OCX spin boxes are too large and oriented in the wrong direction. You need to resize the OCX spin boxes, change their orientation, and add text labels and borders.

Follow these steps to resize the OCX spin boxes and change their orientation:

1 ♦ Use the Window→ Properties Window option in the AppBuilder menu to set the control frames' Heights to **1.52** and their Widths to **9.0**.

See "Creating and Positioning the Buttons" in the "Detailed Customer Section Procedure" section for information on using the Properties Window.

2 ♦ Double-click the left spin box to open the Property Editor - CSSpin dialog box and scroll down to select the Style property:



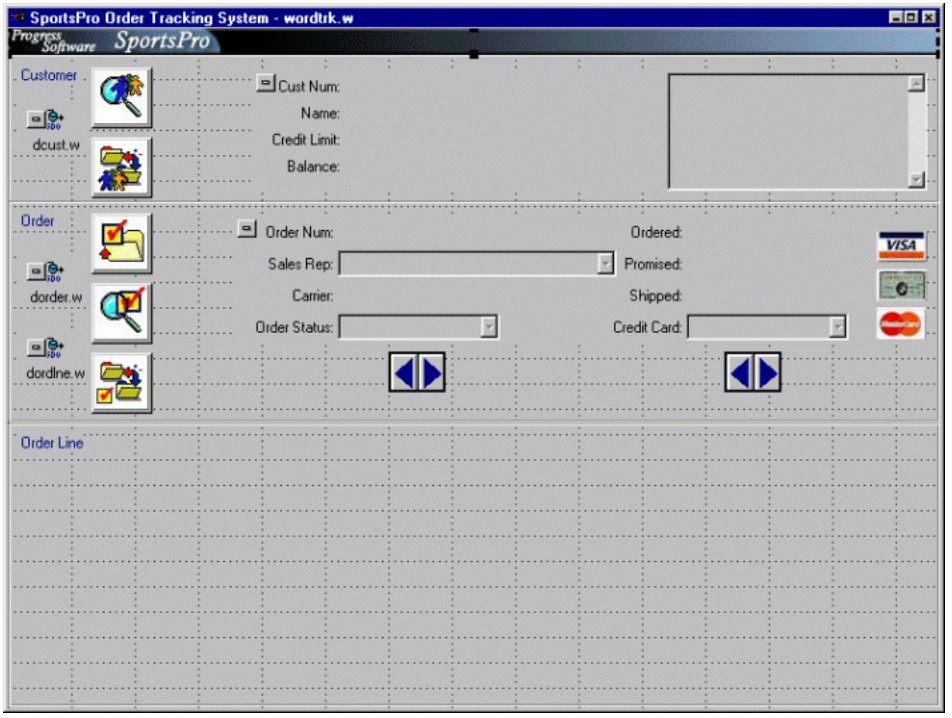
3 ♦ From the Style drop-down list, select 3-3D Horizontal.

4 ♦ From the ArrowColor drop-down list, select a dark blue.

5 ♦ Close the Property Editor dialog box.

6 ♦ Double-click the right spin box and repeat the procedure from Step 2 for the right spin box.

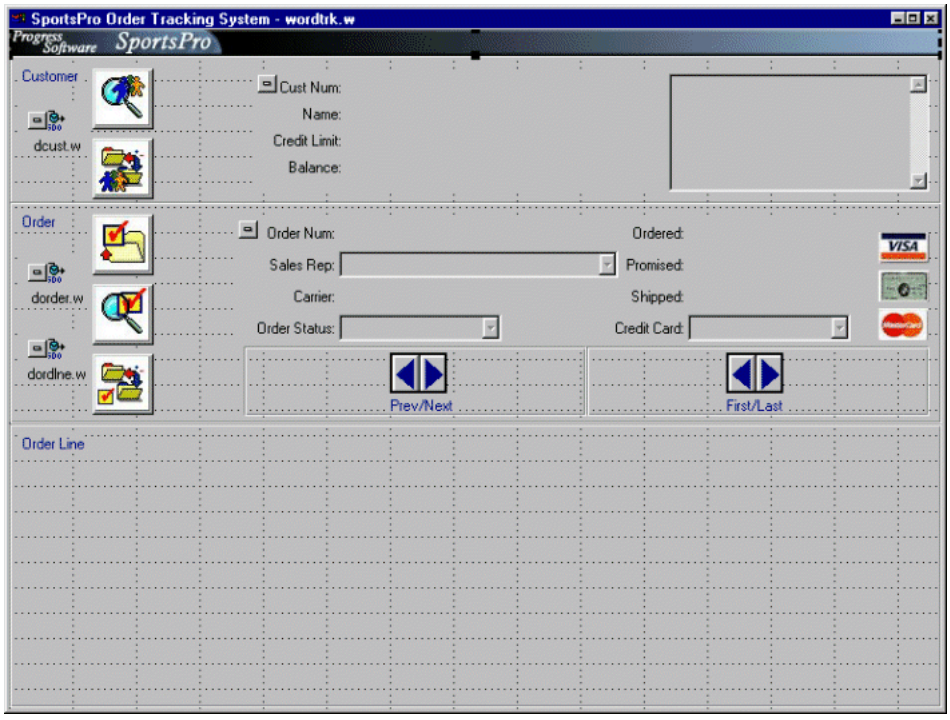
Your window should now look something like this:



7 ♦ Use the Text icon T in the Object Palette to add the colored text labels, **Prev/Next**, below the left spin box and **First/Last** below the right spin box. To review this procedure, see "Adding Text Labels To the Sections" in the "Detailed Customer Section Procedure" section.

8 ♦ Position the left spin box and label so that they are centered under the Order Status control and the right spin box and label so that they are centered under the Credit Card control.

9 ♦ Use the Rectangle icon  in the Object Palette to add a rectangle around each spin box:

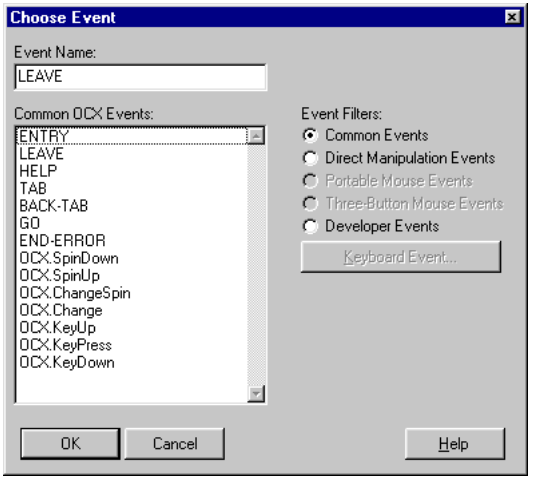


Coding the Prev/Next Spin Box To Fetch the Previous Record

Follow these steps to add trigger code to select the previous record value with the Prev/Next spin box:

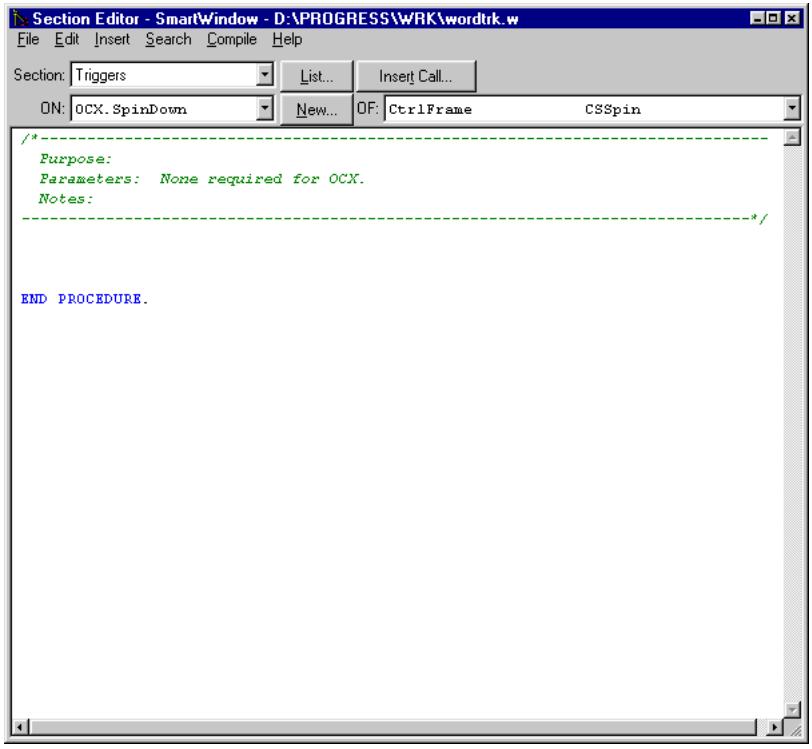
1 ♦ Select the Prev/Next spin box and open the Section Editor.

2 ♦ Choose the New button in the Section Editor to open the Choose Event dialog box:



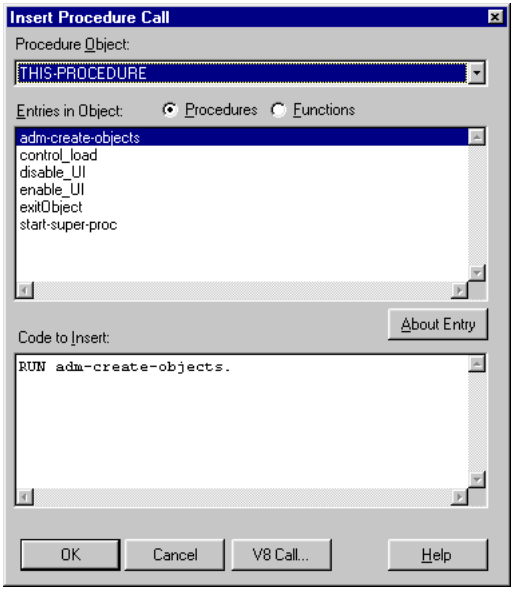
3 ♦ Select OCX.SpinDown from the Common OCX Events list.

The Section Editor now displays the new code section:



4 ♦ Position the cursor below the comments section and above the END PROCEDURE statement.

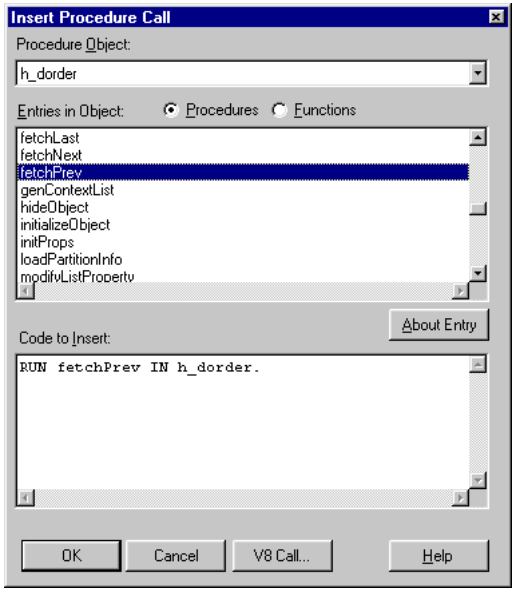
5 ♦ Choose the Insert Call button to open the Insert Procedure Call dialog box:



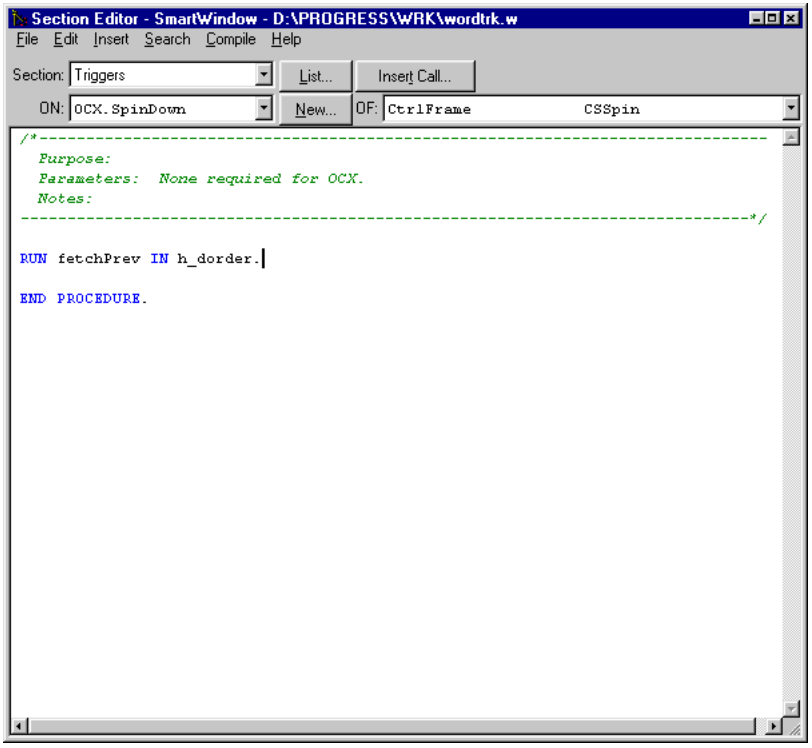
6 ♦ Select `h_dorder` from the Procedure Object drop-down list.

7 ♦ Select the Procedures radio button in the Entries in Object section, and select fetchPrev.

The Code to Insert section shows a preview of the code for this procedure call:



8 ♦ Choose OK. The new code appears in the Section Editor:



Coding the Prev/Next Spin Box To Fetch the Next Record

Follow these steps to add trigger code to select the next record value with the Prev/Next spin box:

1 ♦ Choose New in the Section Editor to open the Choose Event dialog box.

2 ♦ Select OCX.SpinUp in the Common OCX Events list.

3 ♦ Position the cursor below the comments section and above the END PROCEDURE statement.

4 ♦ Choose the Insert Call button to open the Insert Procedure Call dialog box.

5 ♦ Select `h_dorder` from the Procedure Object drop-down list.

6 ♦ Select the Procedures radio button in the Entries in Object section, and select fetchNext.

7 ♦ Choose OK.

The new trigger code appears in the Section Editor.

8 ♦ In the Section Editor, choose File→ Close Window to save the new code and close the window.

Coding the First/Last Spin Box To Fetch the First Record

Follow these steps to add trigger code to select the first record value with the First/Last spin box:

1 ♦ Select the First/Last spin box and open the Section Editor.

2 ♦ Choose the New button in the Section Editor to open the Choose Event dialog box.

3 ♦ Select OCX.SpinDown in the Common OCX Events list, and choose OK.

4 ♦ Position the cursor below the comments section and above the END PROCEDURE statement.

5 ♦ Choose the Insert Call button to open the Insert Procedure Call dialog box.

6 ♦ Select `h_dorder` from the Procedure Object drop-down list.

7 ♦ Select the Procedures radio button in the Entries in Object section, and select fetchFirst.

8 ♦ Choose OK.

The new trigger code appears in the Section Editor.

Coding the First/Last Spin Box To Fetch the Last Record

Follow these steps to add trigger code to select the last record value with the First/Last spin box:

1 ♦ Choose the New button in the Section Editor to open the Choose Event dialog box.

2 ♦ Select OCX.SpinUp in the Common OCX Events list, and choose OK.

3 ♦ Position the cursor below the comments section and above the END PROCEDURE statement.

4 ♦ Choose the Insert Call button to open the Insert Procedure Call dialog box.

5 ♦ Select `h_dorder` from the Procedure Object drop-down list.

6 ♦ Select the Procedures radio button in the Entries in Object section, and select fetchLast.

7 ♦ Choose OK.

The trigger code appears in the editor.

8 ♦ In the Section Editor, choose File→ Close Window to save the new code and close the window.

Viewing the Window

You have now completed assembling the Customer and Order sections of the application's main window. If you want to view and test the partially assembled application, run the application to view the window at this point in the assembly. Before you run the application, save your work.

Follow these steps to view the partially assembled window:

1 ♦ Choose the Run icon  on the AppBuilder toolbar. Your window should appear as it would in a running application. The main window of the sample application should look like the following window:



2 ♦ Hold your pointer over a button to view the ToolTip.

3 ♦ Choose the context sensitive help button (?) in the upper-right corner of the window and then choose a button to see the context sensitive help.

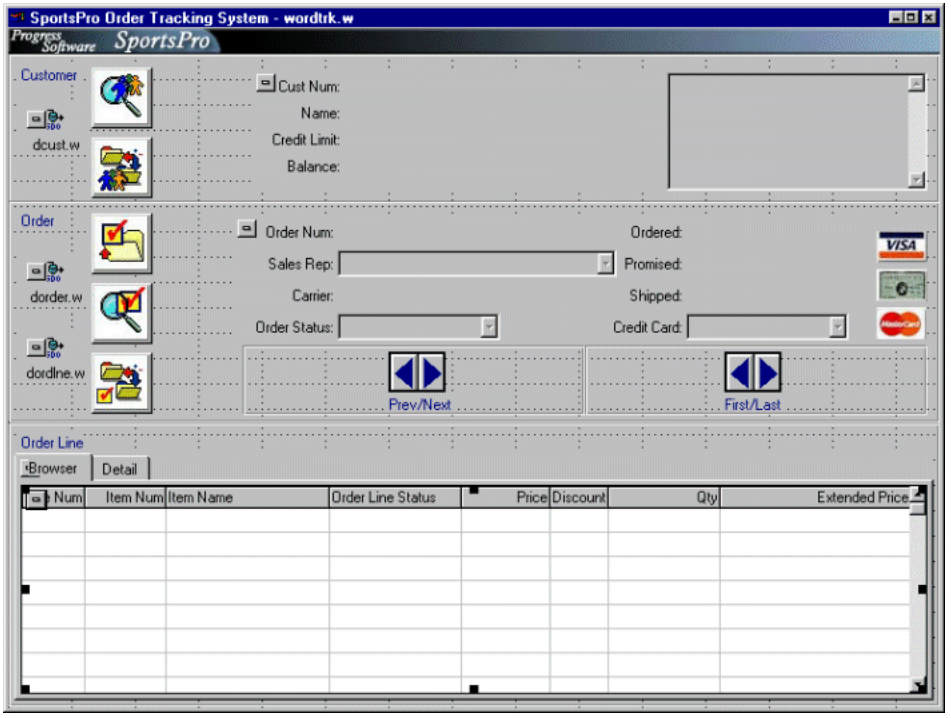
4 ♦ Choose one of the buttons to launch another module.

5 ♦ Choose the Stop icon  on the AppBuilder toolbar to return to the main window.

Now you can move on to building the Order Line section.

3.6 Building the Order Line Section

The Order Line section contains a SmartFolder with SmartObjects for viewing and browsing order line information. When you are finished with this section of the tutorial, your window should look like this:



3.6.1 Task List and Short Procedure

To reach this point, you need to know how to do the following tasks:

1. Add a SmartFolder.
2. Link the SmartFolder to the SmartWindow.
3. Define two tabs on the SmartFolder.
4. Set logical pages.
5. Add a prebuilt SmartDataBrowser (SDB) to Page 1.
6. Link an SDB to an SDO.
7. Add a prebuilt SDV to Page 2.
8. Link an SDV to an SDO.
9. Add a Navigation SmartPanel to Page 2.
10. Link the SmartPanel to the an SDO.

The following short procedure outline how to complete these tasks:

1 ♦ Add a SmartFolder to the Order Line section.

To review this procedure, see the "Adding a SmartFolder" section.

2 ♦ Create a Page SmartLink from the SmartFolder to the SmartWindow.

To review this procedure, see Step 2 in the "Adding a SmartFolder" section.

3 ♦ Resize the SmartFolder.

To review this procedure, see Step 4 in the "Adding a SmartFolder" section.

4 ♦ Create Browser and Detail tabs for the SmartFolder.

To review this procedure, see Step 5 in the "Adding a SmartFolder" section.

5 ♦ Assign a logical page to the Browser tab and set it to display on startup.

To review this procedure, see the "Creating Page 1 Of the SmartFolder" section.

6 ♦ Add the prebuilt SDB, `bordlne2.w`, to the Browser tab.

To review this procedure, see the "Adding a SmartDataBrowser to Page 1 Of the SmartFolder" section.

7 ♦ Create a Data SmartLink from the SDO, `dordlne.w`, to the SDB.

To review this procedure, see Step 3 in the "Adding a SmartDataBrowser to Page 1 Of the SmartFolder" section.

8 ♦ Create an Update SmartLink from the SDB to the SDO, `dordlne.w`.

To review this procedure, see Step 4 in the "Adding a SmartDataBrowser to Page 1 Of the SmartFolder" section.

9 ♦ Add the prebuilt SDV, `vordlne.w`, to the Details tab.

To review this procedure, see the "Adding a SmartDataViewer to Page 2 Of the SmartFolder" section.

10 ♦ Create a Data SmartLink from the SDO, `dordlne.w`, to the SDV.

To review this procedure, see Step 5 in the "Adding a SmartDataBrowser to Page 1 Of the SmartFolder" section.

11 ♦ Add a Navigation SmartPanel to the Details tab.

To review this procedure, see the "Adding a Navigation SmartPanel" section.

12 ♦ Create a Navigation SmartLink from the SmartPanel to the SDO, `dordlne.w`.

To review this procedure, see Step 3 in the "Adding a Navigation SmartPanel" section.

3.6.2 Detailed Order Line Section Procedure

The following sections lead you through assembling the Order Line section in more detail.

Adding a SmartFolder

In this section, you:

- Add a SmartFolder with tabs to the SmartWindow.
- Define a logical page for each tab.
- Add SmartObject to the tabs.

In the Order Tracking System main window, the SmartWindow uses three logical pages (0, 1, and 2) to display SmartObjects. Logical pages control which SmartObjects are displayed. Objects on the default logical page, page 0, are always visible. The SmartFolder uses logical pages 1 and 2 for its tabs. The SmartFolder controls the display of SmartObjects on these pages by toggling their visibility depending on which tab is selected.

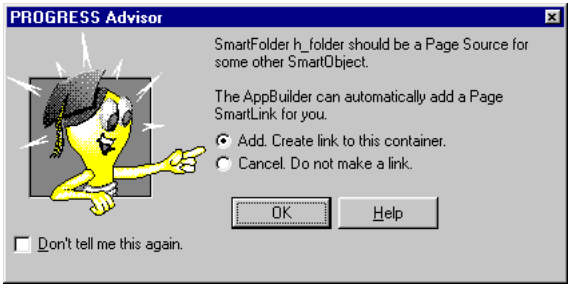
NOTE: A SmartFolder can only control the display of SmartObjects. A basic object, such as a text object or a rectangle, must be included in a SmartObject before a SmartFolder can control its display.

Follow these steps to add a SmartFolder:

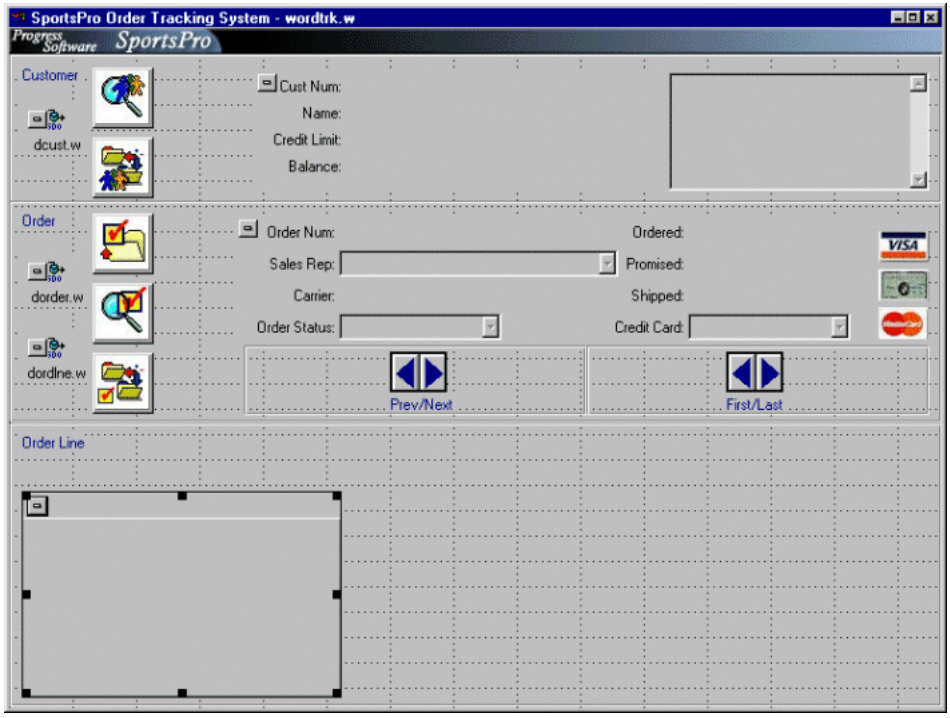
1 ♦ Choose the SmartFolder icon  in the Object Palette.

2 ♦ Position the pointer near the left edge of the Order Line section in the window and click once.

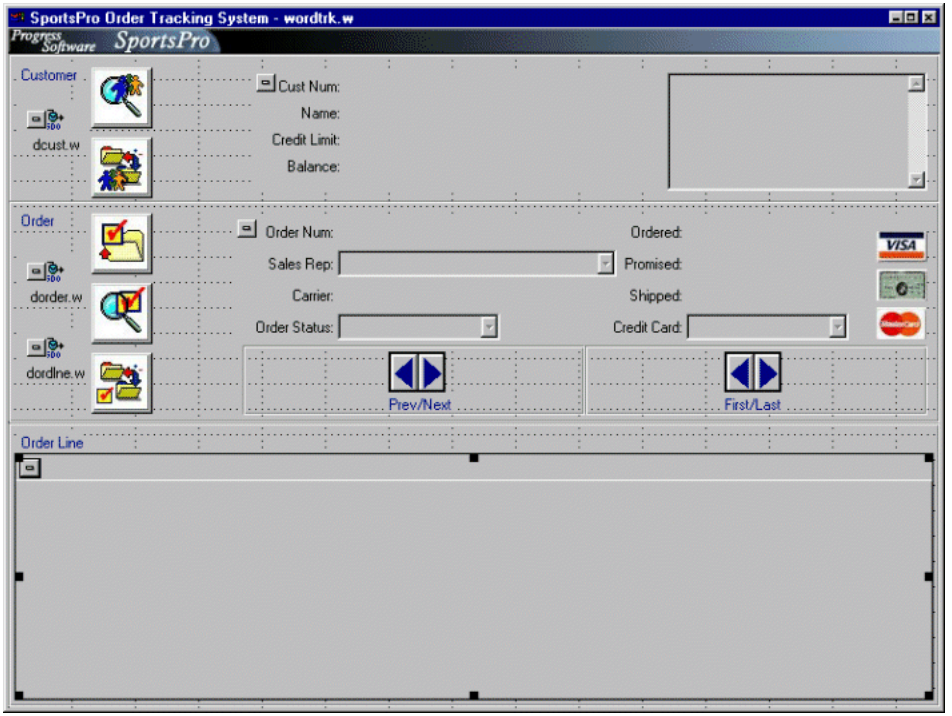
The Progress Advisor opens, notifies you that the SmartFolder should be a Page Source for some other SmartObject, and offers to create a Page SmartLink to the container (the SmartWindow):



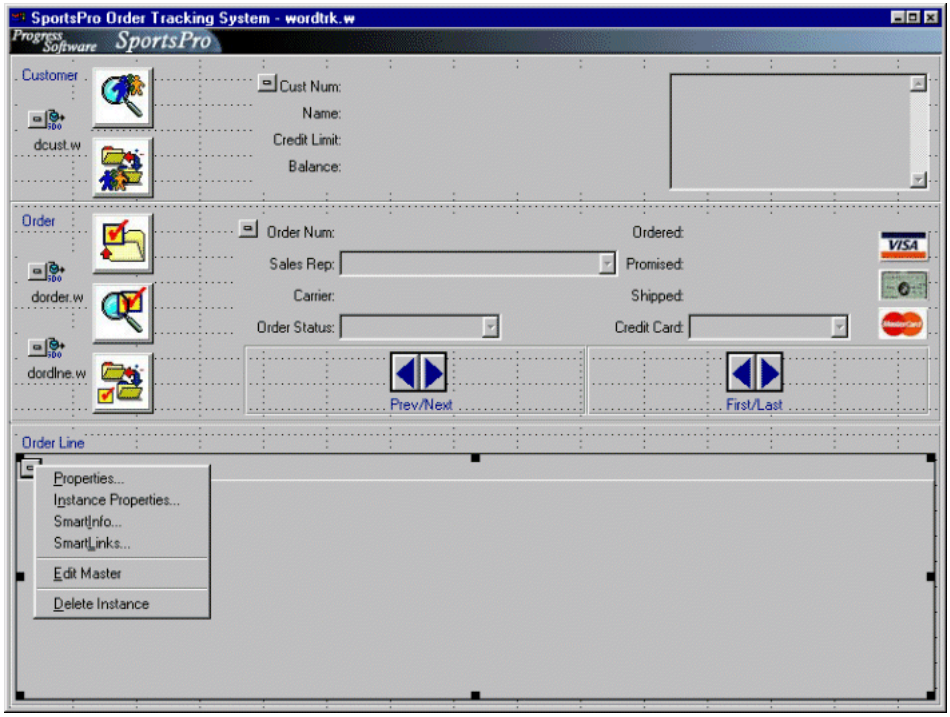
3 ♦ Choose OK and the SmartFolder appears in the design window:



4 ♦ Resize and reposition the folder, as shown in the following figure:



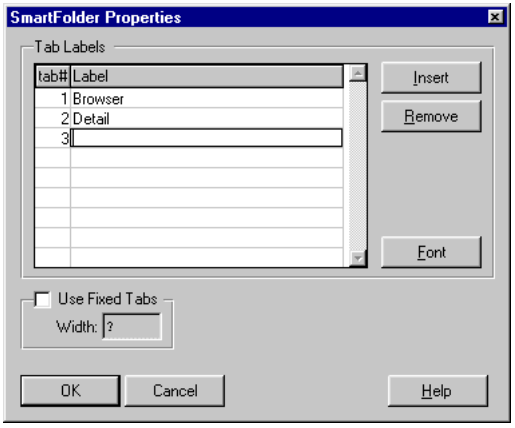
5 ♦ With the right mouse button, open the SmartObject Instance menu, the square icon in the upper-left corner of the SmartFolder, and select Instance Properties from the pop-up menu:



6 ♦ In the SmartFolder Properties dialog box, insert the following tab labels:

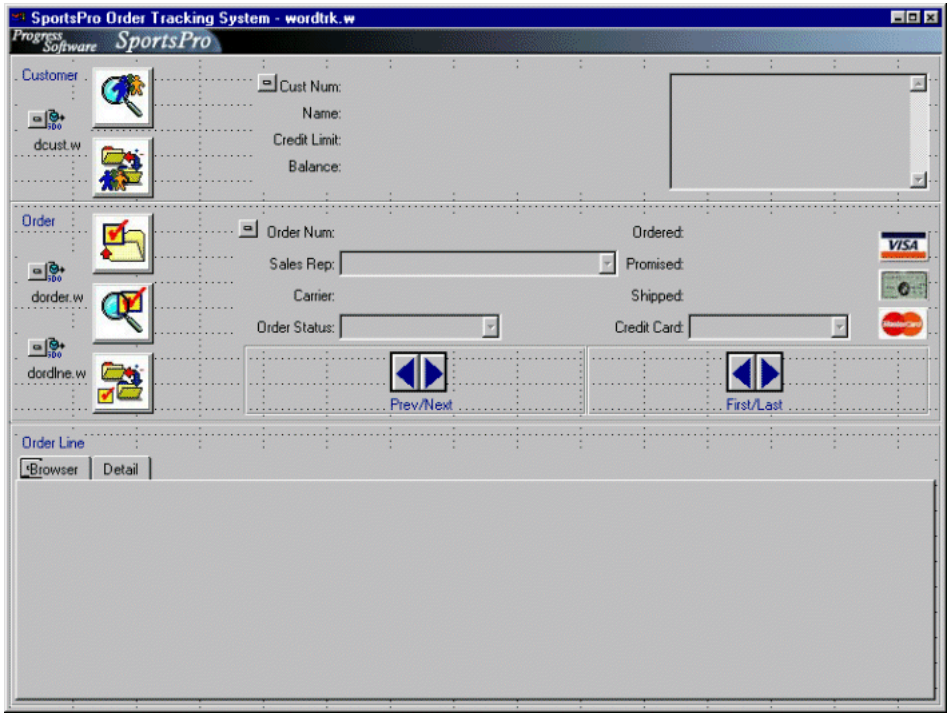
Tab	Label
1	Browser
2	Detail

The completed dialog box should look like this:



7 ♦ Choose OK.

The SmartFolder now contains the two tabs with the labels you entered, as shown in the following figure:



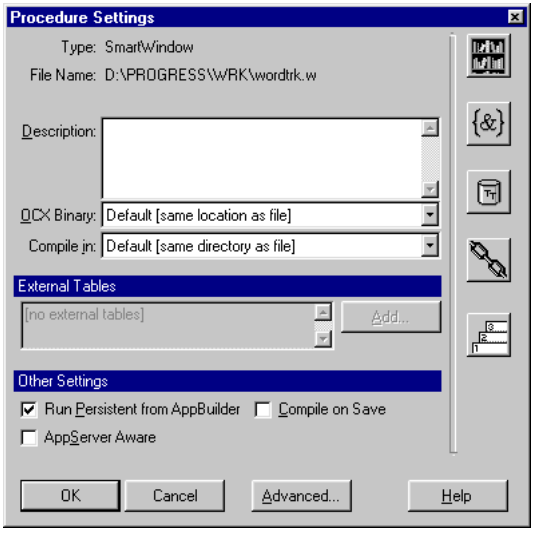
8 ♦ Save your work.

Creating Page 1 Of the SmartFolder

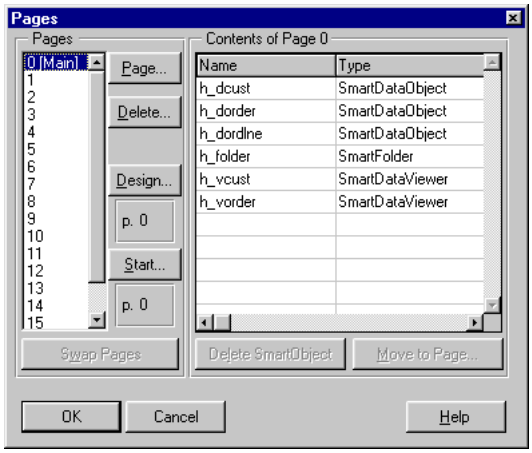
Before you can add objects to the SmartFolder tabs, you must define logical pages for the tabs. In this instance, the Browser tab of the SmartFolder controls logical Page 1. When you add a logical page for a tab, you control which objects appear when that tab is selected.

Follow these steps to define logical pages for the SmartFolder:

1 ♦ Choose the Procedure Settings icon  in the AppBuilder toolbar to open the Procedure Settings dialog box:

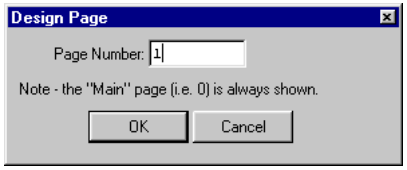


2 ♦ Choose the Pages icon  to open the Pages dialog box:



The page numbers begin with Page 0. Page 0 is the default (Main) page. Objects assigned to Page 0 are always displayed, regardless of which folder tab is selected. The objects that you added to the main window so far, such as `dcust`, `dorder`, and the SmartFolder itself, are all assigned to Page 0 by default.

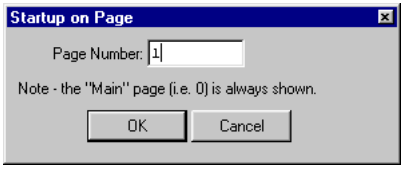
3 ♦ Choose Design to open the Design Page dialog box:



4 ♦ Set the Page Number field to **1** and choose OK.

Page 1 is now the current design page and is assigned to the Browser tab. Any objects that you now add to the SmartWindow display only when the Browser tab is selected.

5 ♦ Choose the Start button on the Pages dialog box to open the Startup on Page dialog box:



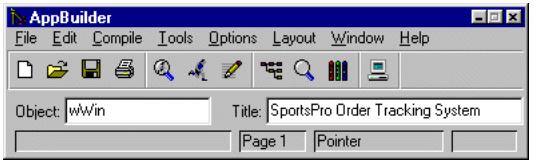
The value in the Page Number field determines which page, if any, displays in addition to Page 0 when the application first runs. In the SportsPro Order Tracking System, you want Page 1 to be the startup page so the SDB displaying the order lines opens at startup.

6 ♦ Set the Page Number field to **1** and choose OK.

Page 1 now displays at startup, along with the main page.

7 ♦ Commit your changes to return to the main window.

The Status Area of the AppBuilder indicates you are now working in Page 1:



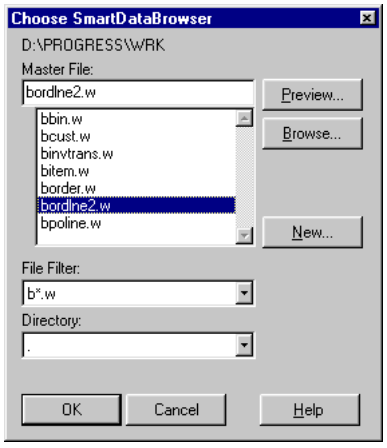
The SmartFolder is now configured to control the display of other SmartObjects.

Adding a SmartDataBrowser to Page 1 Of the SmartFolder

Your next task is to add an SDB to Page 1, the Browser tab.

Follow these steps to add an SDB:

1 ♦ From the Object Palette, choose the SDB icon  to open the Choose SmartDataBrowser dialog box:



2 ♦ Select `bordlne2.w` and choose OK.

3 ♦ Click within the SmartFolder.

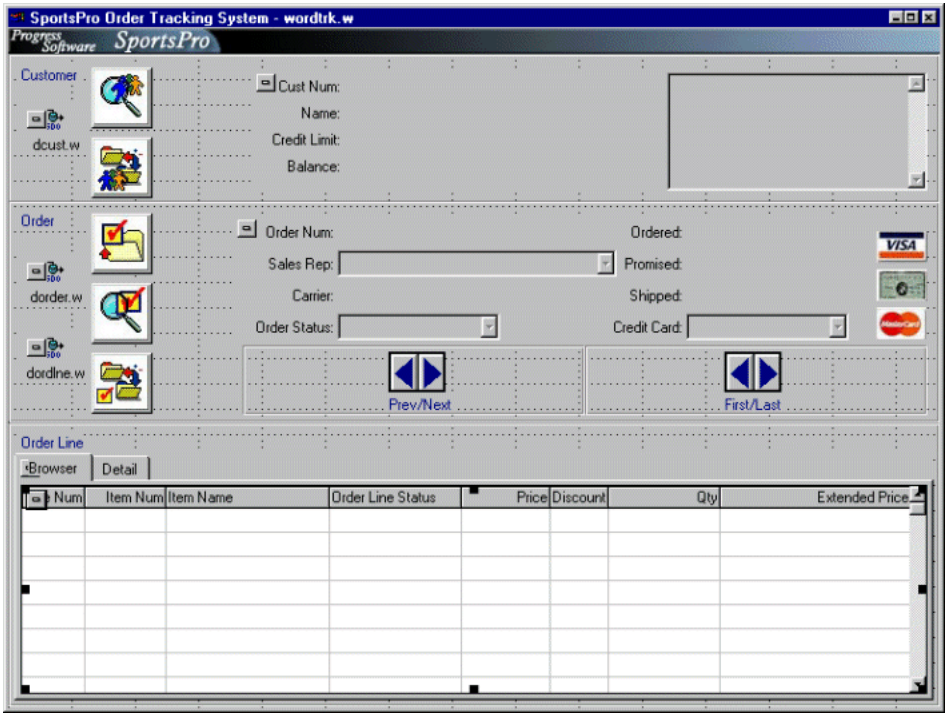
The SDB appears in the SmartFolder. The Progress Advisor opens, notifies you that the SDB instance, `h_bordlne2`, should be a Data Target for some other SmartObject, and offers to create a Data SmartLink from the SDO instance, `h_dordlne`.

4 ♦ Select the radio button to create the SmartLink and choose OK.

The Progress Advisor opens, notifies you that the SDB instance, `h_bordlne2`, should be an Update Source for some other SmartObject, and offers to create an Update SmartLink to the SDO instance, `h_dordlne`.

5 ♦ Select the radio button to create the SmartLink and choose OK.

6 ♦ Position the SDB in the SmartFolder, as shown in the following figure:



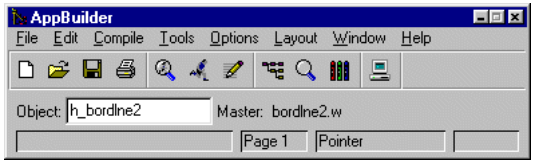
Now that you have built the Browser tab, you are ready to build the Detail tab.

Adding a SmartDataViewer to Page 2 Of the SmartFolder

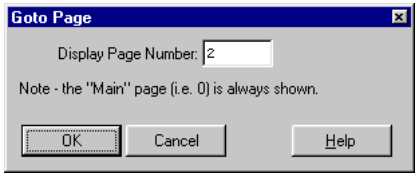
Before you add the SDV, `vordlne`, you must reset the current design page.

Follow these steps to specify design page 2 and add an SDV:

1 ♦ Double-click the Page Number field in the Status Area of the AppBuilder:



2 ♦ In the Goto Page dialog box, set the Display Page Number field to **2** and choose OK:



The main window opens again with the Detail tab selected.

3 ♦ Choose the SDV icon  in the Object Palette to open the Choose SmartDataViewer dialog box.

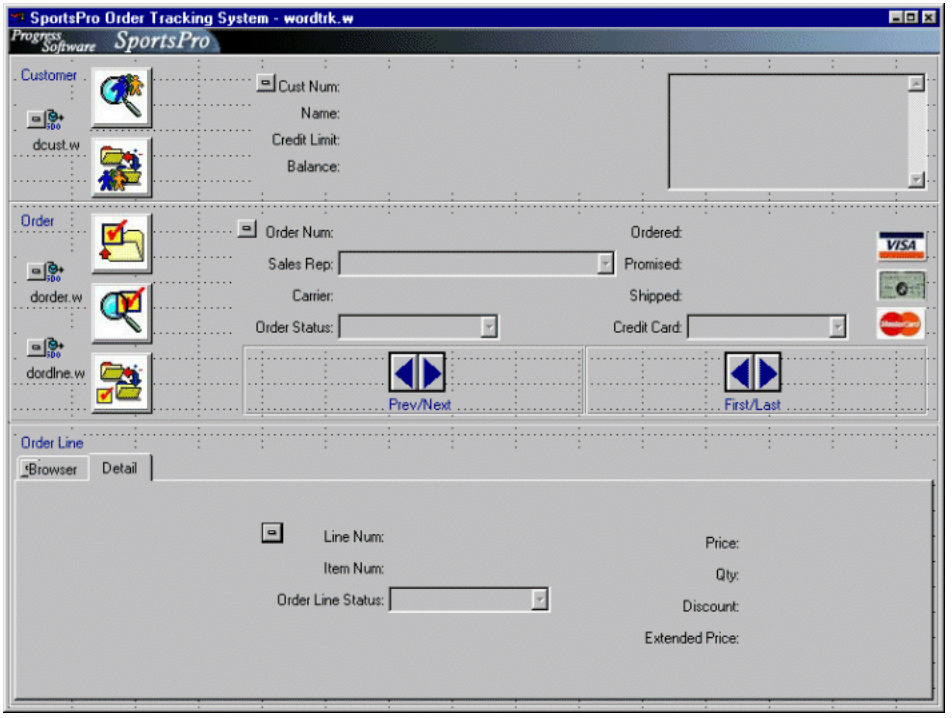
4 ♦ Select `vordlne.w` and choose OK.

5 ♦ Click once in the Detail tab.

The Progress Advisor opens, notifies you that the SDV instance, `h_vordlne`, should be a Data Target for some other SmartObject, and offers to add a Data SmartLink from the SDO instance, `h_dordlne`.

6 ♦ Select the radio button to create the SmartLink and choose OK.

7 ♦ Position the SDV, as shown in the following figure:

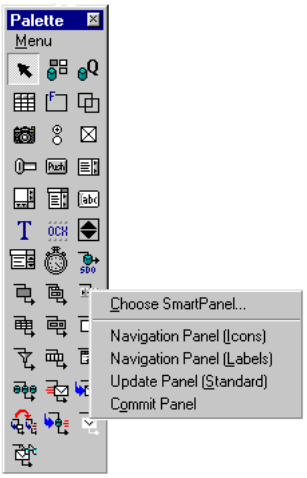


Adding a Navigation SmartPanel

You now add the Navigation SmartPanel, `pnavico`, to the Detail tab.

Follow these steps to add a Navigation SmartPanel:

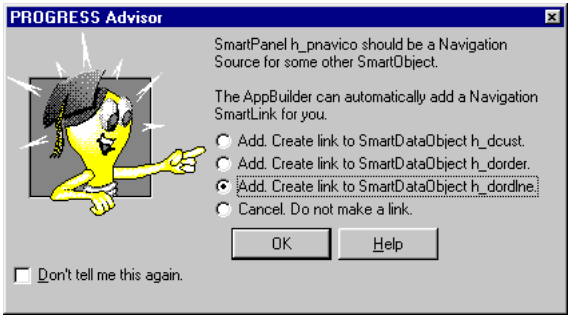
1 ♦ Choose the SmartPanel icon  on the Object Palette and use the right mouse button to open a pop-up menu:



2 ♦ Select Navigation Panel (Icons) from the menu.

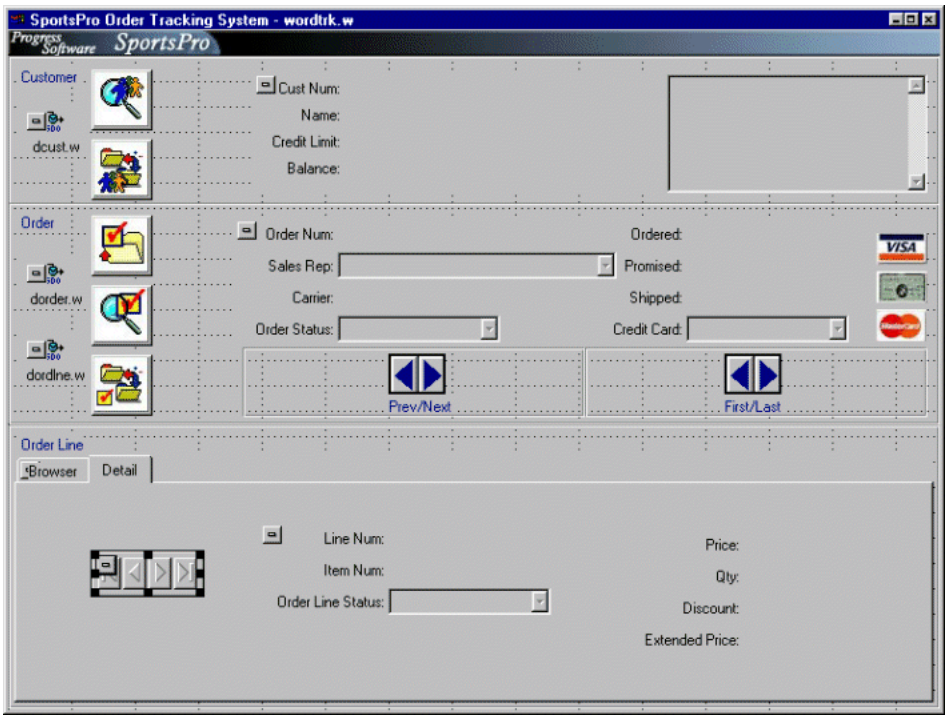
3 ♦ Position the pointer to the left of the SDV, and click once.

The Progress Advisor opens, notifies you that the SmartPanel, `pnavico`, should be a Navigation Source for some other SmartObject, and offers to add a Navigation SmartLink to one of the following SDO instances, `h_dcust`, `h_dorder`, or `h_dordlne`:



4 ♦ Select the radio button to create the SmartLink to `h_dordlne` and choose OK.

Move the SmartPanel into position, as shown in the following figure:



5 ♦ Save your work.

You have now assembled the main window of the SportsPro Order Tracking system using predefined objects. You should have a better understanding of how to use the ADE and AppBuilder to assemble SmartObject applications. Run your application and test the code you created.

If you have not already run the complete sample application, you might want to run it now. Running the application gives you an opportunity to see some of the other capabilities of ADM2 SmartObjects. For information on running the application, see Chapter 2, “Working With the Sample Application.”

If you would like to work more with SmartObjects, you can complete the exercises in Chapter 4, “An Overview Of SmartObject Technology.” These exercises show how to fabricate SmartObject masters from templates installed with the product.

4

An Overview Of SmartObject Technology

The SportsPro sample application's Order Tracking System solves a realistic business problem that a rapidly growing distribution business might encounter. Because of the importance of high-level planning in object-based application development, it is important to design your application and to plan how to use SmartObjects in the application.

The chapter starts with a brief introduction to SmartObjects. It then describes the design and development phases necessary for creating an application and how the design translates into the required set of SmartObjects and SmartLinks. Finally, it presents some exercises that you can complete to fabricate a SmartDataObject, SmartDataViewer, and SmartDataBrowser using SmartObject templates included with the product. Using existing templates to fabricate new SmartObjects helps you understand the relationship among SmartObjects and the simplicity with which you can create new SmartObjects to build an application.

This chapter covers the following sections:

- SmartObject Technology
- SmartObjects Used In the Order Tracking System
- Business Scenario
- Development Phases
- Building SmartDataObjects
- Building SmartDataViewers
- Building SmartDataBrowsers

4.1 SmartObject Technology

The SmartObjects in the main window of the sample application are based on SmartObject templates. Templates provide a starting point for creating masters. At run time, you run an instance of a master, and this instance is a SmartObject. Progress provides a starter set of SmartObject templates from which to build your own masters for creating end-user applications. For example, in the SportsPro Order Tracking System, the SmartObject named bcust is a master based on the SmartDataBrowser template.

SmartObjects provide a significant advantage in that you can reuse them. When you use a SmartObject master in an application, the AppBuilder does not write its actual code into the window file it creates; it only writes instructions to create an instance of the master. You can have many applications that use instances of a single master, and you usually use multiple instances of the same master in one application.

SmartObjects are specialized 4GL procedures that run *persistently*. After a SmartObject performs a specific task, in response to an end-user action, it remains in memory, ready to respond to end-user input and instructions it receives from other SmartObjects. The specific functionality of a SmartObject is encapsulated within a framework of *internal procedures* (also known as methods). Internal procedures are procedure blocks defined inside the context of an external procedure. Some internal procedures are common to all SmartObjects, and some give each standard type of SmartObject its unique characteristics. For more information about internal and external procedures, see the *Progress AppBuilder Developer's Guide*, the *Progress ADM 2 Guide*, and the *Progress ADM 2 Reference*.

Using SmartObjects in applications makes even the most complex applications easy to maintain. Since all instances of a SmartObject master are based on one file, any change you make to that source file applies to every application that uses that master.

4.2 SmartObjects Used In the Order Tracking System

In Chapter 2, "Working With the Sample Application" and Chapter 3, "Assembling the Order Tracking System," you work with the SportsPro sample application to better understand the SmartObject technology and how to design and create an application. Figure 4–1 shows the main window of the SportsPro Order Tracking System as it appears when you run it. For information on running the sample application, see the "Running the Sample Application" section in Chapter 2, "Working With the Sample Application."

Figure 4–1 shows the main window of the sample application and identifies some of the SmartObjects in this window.

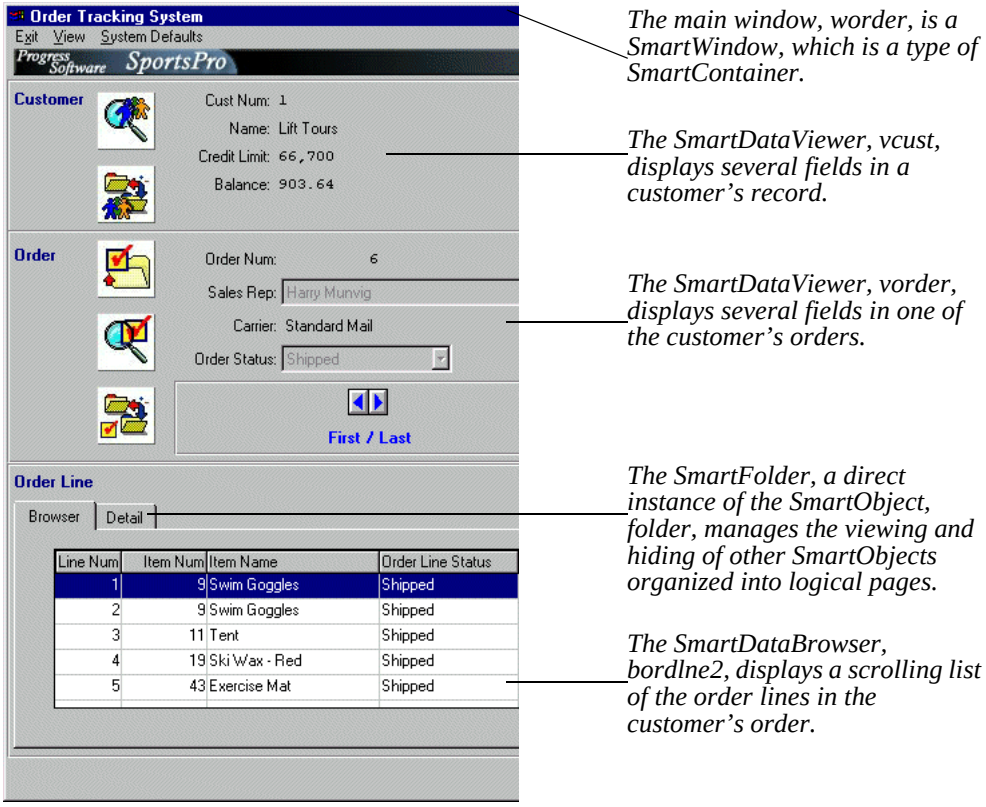


Figure 4–1: SmartObjects In the Order Tracking System

Some SmartObjects used in the Order Tracking System above include:

- **SmartDataObjects** — Sources for Data SmartLinks that target SmartDataViewers and other SmartDataobjects
- **SmartFolder** — Source for a Page SmartLink that is targeted to a SmartWindow to control paging in that window
- **SmartDataLinks** — Establish a communication link from SmartDataObjects to SmartDataBrowsers and SmartDataViewers
- **SmartDataViewer** — Displays data from SmartDataObjects
- **SmartDataBrowser** — Displays data from SmartDataObjects and allows a user to scroll through the data

- **SmartPanel** — Contains buttons that control navigation as a source of a Navigation SmartLink through a SmartDataObject's query
- **Navigation SmartLink** — Establishes a navigation SmartLink from a SmartPanel to a SmartDataObject

In addition to the objects listed above, the main window uses a SmartContainer. A SmartContainer is a special kind of SmartObject designed to contain other SmartObjects (including possibly other SmartContainers). Progress supplies three types of SmartContainers:

- SmartWindows
- SmartDialogs
- SmartFrames

Each SmartContainer type keeps track of all of the information required to create running instances of the SmartObjects it contains. This information includes the positions of the SmartObjects and how they are linked.

The SmartObjects in the main window illustrate different types of SmartObjects. How a SmartObject functions in an application depends on the type of SmartObject and the SmartLinks defined for the SmartObject. SmartLinks are pathways used to send messages among SmartObjects.

When you assemble the main window of the sample application in Chapter 3, "Assembling the Order Tracking System," you used the Progress Advisor to create SmartLinks for the SmartObjects you added.

Figure 4–2 shows the different types of SmartObjects used in the main window of the sample application, identifies the file, and shows SmartLinks for these SmartObjects. For more information about files, see the "Naming Conventions" section in this chapter.

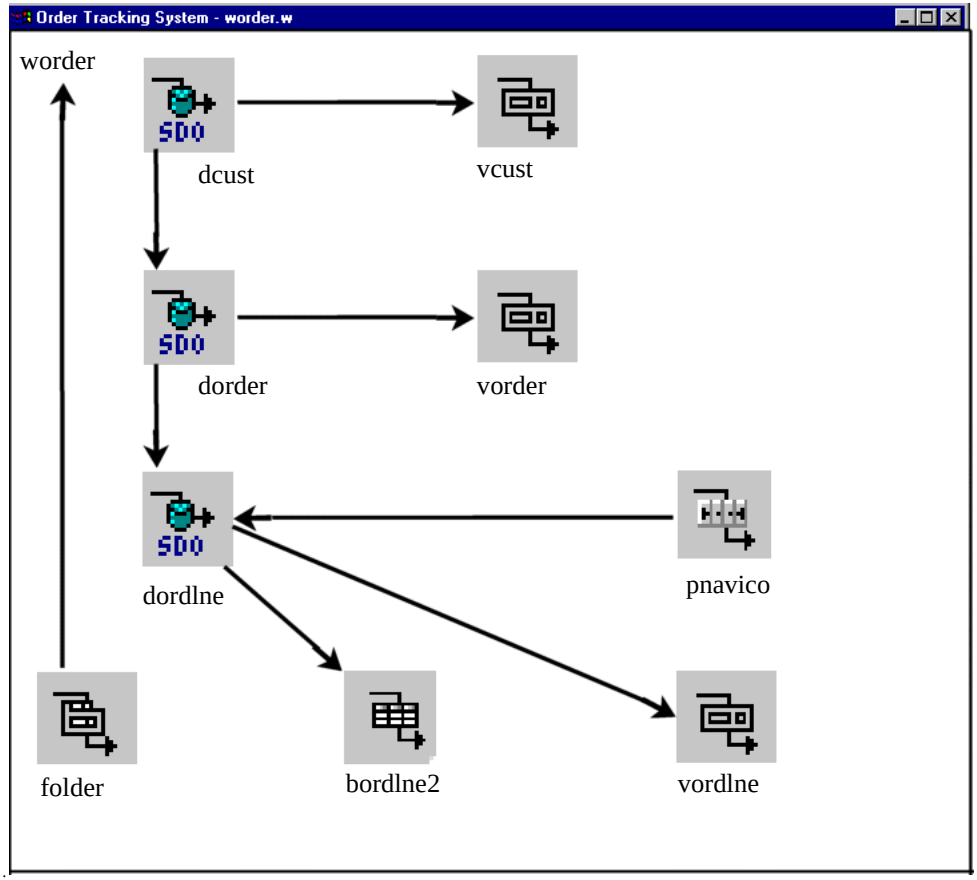


Figure 4–2: SmartObject Types and SmartLinks

A SmartObject does not know how it relates to other SmartObjects until you create the links. You determine the relationship by using a particular SmartLink type and by specifying which end of the link each object is on (Source or Target). For all SmartLink types, a Source initiates some action or passes some data to one or more Targets. In general, a Source can have numerous Targets simultaneously, but a Target has only one Source at a time. For more information about SmartLinks, see the *Progress AppBuilder Developer's Guide*.

SmartLinks provide direct communication pathways between pairs of SmartObjects. A single SmartObject relates to other SmartObjects in a variety of ways. For example, a SmartDataObject relates to a SmartDataViewer differently than it relates to a Navigation SmartPanel. Most SmartObjects support a variety of SmartLink types. Some, like the Navigation SmartPanel, only support one SmartLink type (Navigation).

SmartLinks are directional in nature. Each SmartLink is directed from a source SmartObject to a target SmartObject. There are several types of SmartLinks. For example, a Data SmartLink sends the contents of a database record, while a Navigation SmartLink sends navigation instructions.

The following figures group the SmartObjects shown in Figure 4–2 based on their SmartLinks. The figures describe the types of SmartLinks and the different types of SmartObjects.

Figure 4–3 shows that the SmartFolder, folder, is the source for a Page SmartLink targeted to the SmartWindow, `worder`. The Page SmartLink tells the main window which logical page to display.

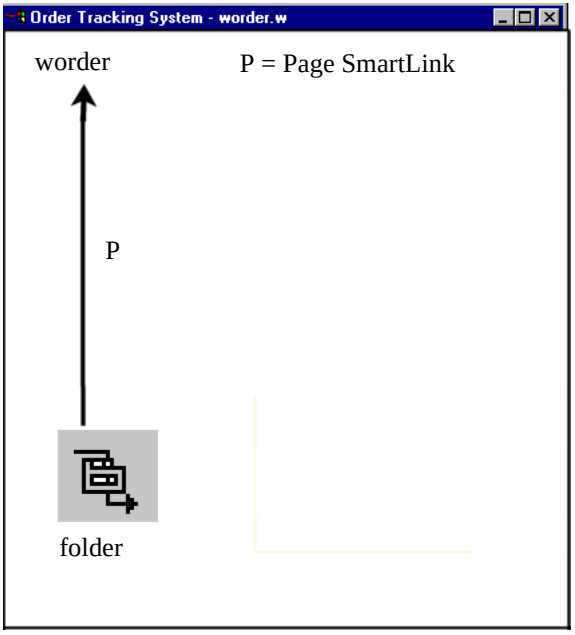


Figure 4–3: SmartFolder and Page SmartLink

Figure 4–4 shows that the SmartDataObject, `dcust`, is the source for two Data SmartLinks targeted to the SmartDataViewer, `vcust`, and the SmartDataObject, `dorder`.

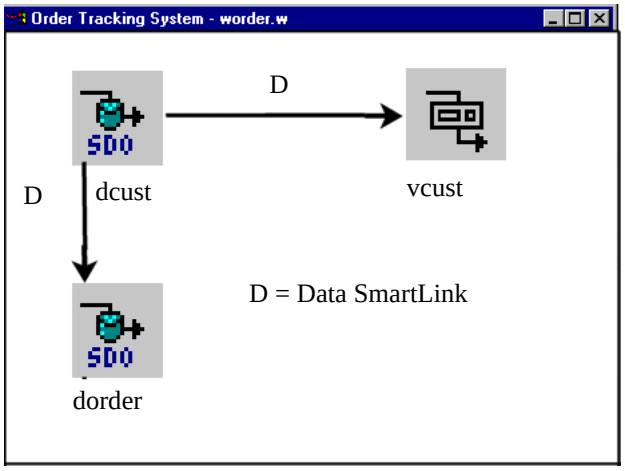


Figure 4–4: SmartLinks For dcust SDO

Figure 4–5 shows that the SmartDataObject, `dorder`, is the source for two Data SmartLinks targeted to the SmartDataViewer, `vorder`, and the SmartDataObject, `dordlne`.

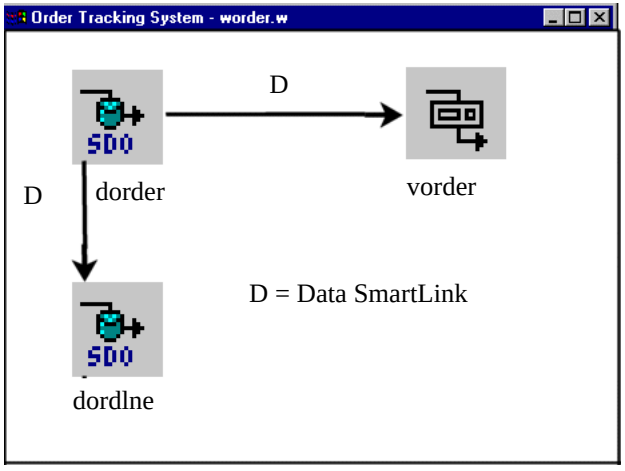


Figure 4–5: SmartLinks For dorder SDO

Figure 4–6 shows that the SmartDataObject, `dordlne`, is the source for two Data SmartLinks targeted to the SmartDataBrowser, `bordlne2`, and the SmartDataViewer, `vordlne`. The SmartPanel, `pnavico`, is the source for a Navigation SmartLink to the SmartDataObject, `dordlne`.

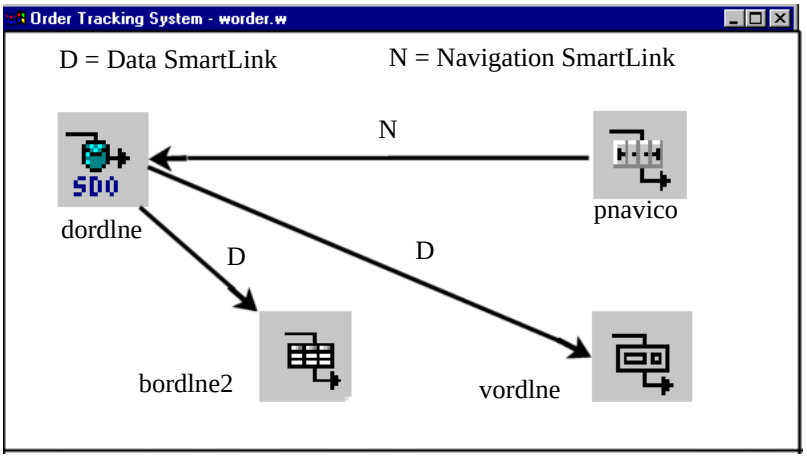


Figure 4–6: SmartLinks For dordlne SDO

4.2.1 Naming Conventions

The AppBuilder creates structured procedure files, called *window files*, that are identified by a `.w` extension. For example, the source filename of the SmartDataBrowser referred to as `bcust` has a `.w` extension (`bcust.w`). It has a `.r` extension in its compiled form (`bcust.r`) and a prefix of h_ (for handle) when referred to as an instance in memory (`h_bcust`).

Other files commonly used with the AppBuilder are: include files (with a `.i` extension), procedure files (with a `.p` extension), and client proxy files (with a suffix of _cl added to the filename).

Table 4–1 lists the AppBuilder filename conventions.

Table 4–1: AppBuilder Filename Conventions

If a filename begins with...	Then the file is a...
b	SmartDataBrowser
d	SmartDataObject
g	SmartDialog
p	SmartPanel
sbo	SmartBusinessObject
smr	Smart[Message]Receiver
sms	Smart[Message]Sender
sbb	SmartB2BObject
sdf	SmartDataField
v	SmartDataViewer
w	SmartWindow

4.3 Business Scenario

The architecture of the SportsPro Order Tracking System reflects a possible solution to a hypothetical but realistic business problem. Imagine that you are the general manager of a growing sporting goods distribution company. You find that your sales people are spending too much time responding to customers' inquiries about the status of existing orders and too little time generating new orders. Your solution is to institute an order tracking help desk to enter the new orders for the sales people and to field customer inquiries.

You start designing an application that allows users to look up a customer's information as quickly as possible. You want the application structured so the information users need most often is the easiest to access. You determine that users most often need to:

- Enter new orders
- Search for a customer
- Enter a new customer
- Change customer information, such as an address or phone number
- Check inventory quantities
- Search for an order
- Check on the shipping status of an order as a whole
- Retrieve information about specific items that were ordered
- Modify lines in an order

After you know what information you want your application to include, and you create a database that contains the information you need to query for this information, you can begin the development phases.

4.4 Development Phases

To work successfully with SmartObjects, your development organization must work through three phases:

- **Analysis and design phase** — Analyze the types of application screens you want to develop, then break down those screens into the separate components required to build them. In this phase, you identify and design whatever SmartObjects you need.
- **Building phase** — Create the SmartObjects specified in the analysis and design phase.
- **Assembly phase** — Assemble the various application screens by inserting instances of the SmartObjects you create into SmartContainers and then setting the necessary SmartLinks.

4.4.1 Analysis and Design Phase—What SmartObjects Are Needed?

Before you assemble an application screen with SmartObjects, you must analyze how you want the screen to appear and behave. Once you decide what you want a screen to look like and how you want it to behave, you begin to think about building the required SmartObjects.

In most cases, when you build screens, you do not need to design any new types of SmartObjects because you can use Progress-supplied SmartObjects. Consequently, the analysis phase is greatly simplified.

For complex applications, the analysis phase might uncover the need for a new type of SmartObject. You then determine how to design and create the new SmartObject, define its user interface and its behaviors, and write the internal procedures.

4.4.2 Building Phase—Creating the SmartObject Masters

When you know what SmartObjects you want to create, you then begin building SmartObject masters. A SmartObject master is a procedure file that you save on your hard drive and reuse throughout an application. To prepare for the assembly phase, your development organization can build various SmartObjects by using and modifying Progress-supplied SmartObjects, or by defining and creating custom SmartObjects. After building the masters, you insert running instances of them into a SmartContainer and then link them together.

4.4.3 Assembly Phase

After building the SmartObjects masters, you are ready to begin the assembly phase. In this phase, you create a SmartWindow, insert instances of the SmartObjects that you built, and link them to produce a finished, working copy of the application screen.

The rest of this chapter includes exercises for creating a new SmartDataObject, SmartDataViewer, and SmartDataBrowser using prefabricated SmartObject templates. If you have not already read Chapter 2, “Working With the Sample Application” and Chapter 3, “Assembling the Order Tracking System,” you might want to read these chapters before you continue with this chapter. Reading these chapters gives you an understanding of how SmartObjects are used in building applications and should help prepare you for the exercises described in this chapter.

NOTE: Before you begin these exercises, you must be connected to your copy of the Sports2000 database in your working directory. If you need instructions on connecting to the database, see Chapter 2, “Working With the Sample Application.”

4.5 Building SmartDataObjects

In Chapter 3, "Assembling the Order Tracking System," you assembled the main window of the SportsPro Order Tracking System using prefabricated SmartObject masters. In this chapter, the emphasis is on the application development process by showing you how to fabricate SmartObject masters from the standard templates installed with the product.

In this section, you use the SmartDataObject Wizard to create the SmartDataObject, `dcust`. This SmartDataObject (SDO) is one of the SmartObjects used in the main window of the Order Tracking System. SDOs define a set of records to fetch from a database. You cannot see a SDO in a running application. You can only see the information fetched by the SDO when it is displayed in a visualization object such as a SmartDataViewer (SDV) or a SmartDataBrowser (SDB).

In the Order Tracking System, the SDO, `dcust`, in the main window is used to find all rows in the Customer table that match the query entered in the Find Customer dialog box. The SDO also sends the customer record values, one at a time, to the SDV, `vcust`, and the SDO, `dorder`.

The following short procedure outlines the steps in building the SDO:

1 ♦ Start the SmartDataObject Wizard.

To review this procedure, see the "Starting the SmartDataObject Wizard" section.

2 ♦ Define the query.

To review this procedure, see the "Defining the Query" section.

3 ♦ Define the fields.

To review this procedure, see Step 6 in the "Defining the Query" section.

4 ♦ Save the new SDO.

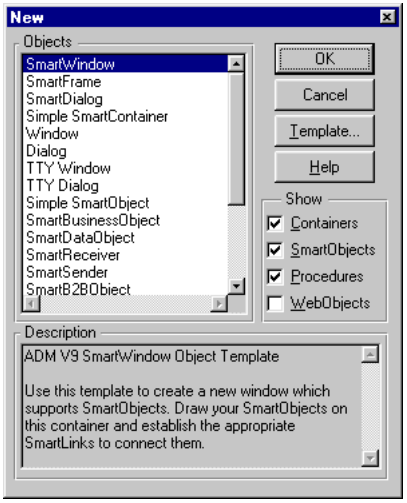
To review this procedure, see the "Saving the New SmartDataObject" section.

4.5.1 Starting the SmartDataObject Wizard

The SmartDataObject Wizard guides you through the steps necessary to create a SDO. The wizard provides information about each step and keeps track of where you are in the process. An especially useful feature of the SmartDataObject Wizard is that it allows you, at any point, to go back and modify work done in earlier steps.

Follow these steps to start the SmartDataObject Wizard:

1 ♦ In the AppBuilder main window, choose the New button to open the New dialog box:



2 ♦ In the New dialog box, select SmartDataObject and choose OK to open Page 1 of the SmartDataObject Wizard:



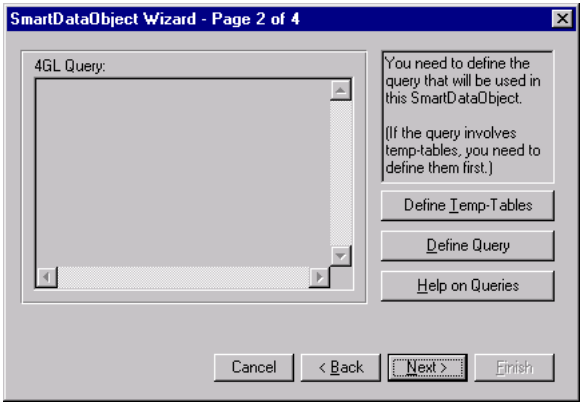
The SmartDataObject Wizard appears similar to the Progress Advisor but wears a different hat. Also notice that the Wizard contains a Back button as well as a Next button. At any time prior to choosing the Finish button, you can use the Back button to return to previous pages to modify your choices.

4.5.2 Defining the Query

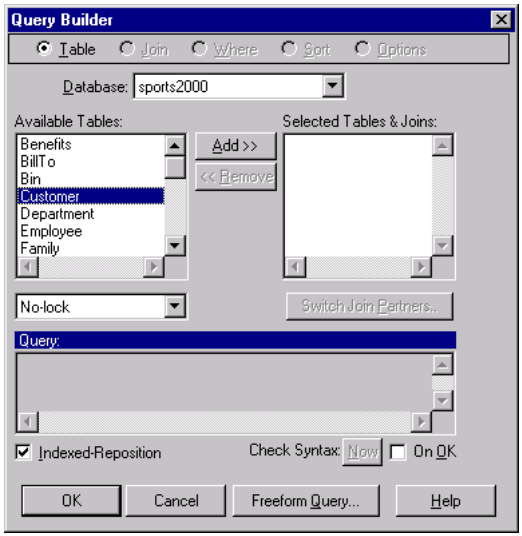
The function of a SDO is to query a database and make the data available to other SmartObjects. These SmartObjects can include SDBs, SDVs, and other SDOs. In this part of the exercise, you are going to select a table and fields from the Sports2000 database for the query.

Follow these steps to define a query:

1 ♦ In Page 1 of the SmartDataObject Wizard, choose Next to open Page 2:



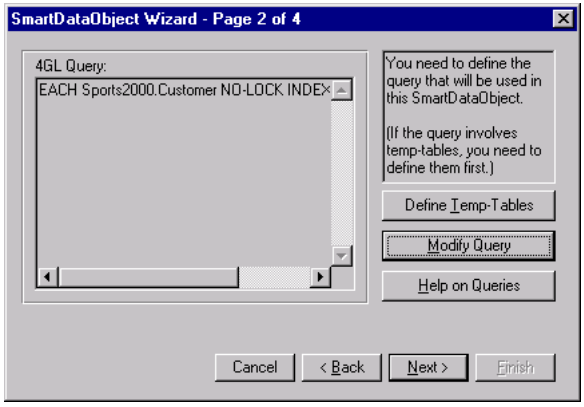
2 ♦ Choose Define Query to open the Query Builder window:



3 ♦ Select Customer from the Available Tables list and choose Add.

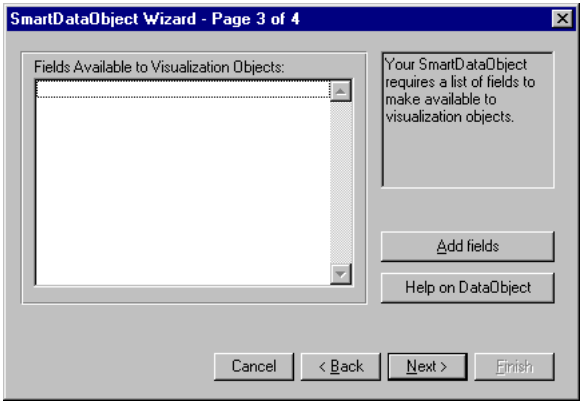
The Customer table now appears in the Selected Tables & Joins selection list, and the 4GL code for the query appears in the Query section.

4 ♦ Choose OK to open Page 2 of the SmartDataObject Wizard:



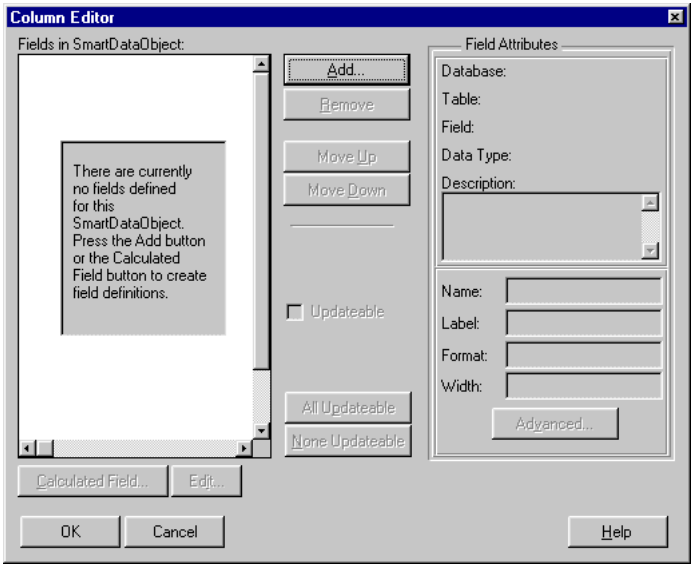
The 4GL Query text box of Page 2 now displays the 4GL code generated by the Query Builder, and the Define Query button label changes to Modify Query. You can use this button to change your query before exiting the SmartDataObject Wizard.

5 ♦ In Page 2 of the SmartDataObject Wizard, choose Next to open Page 3:

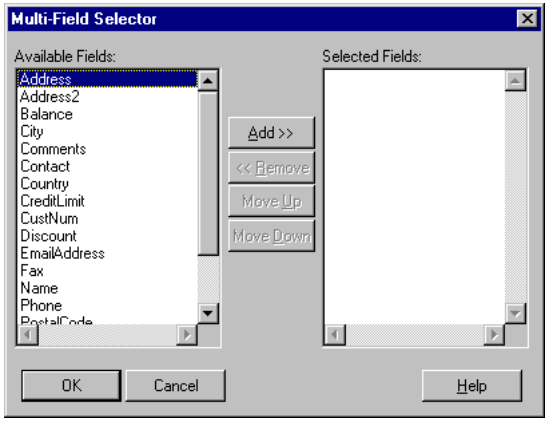


Page 3 is used to define the fields available to visualization objects, such as SDBs and SDVs.

6 ♦ Choose Add fields to open the Column Editor dialog box:



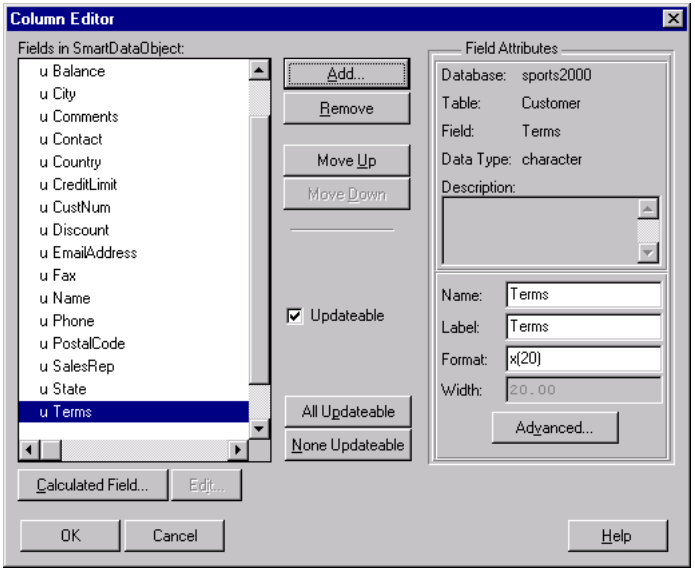
7 ♦ Choose Add to open the Multi-Field Selector dialog box to add fields from the Customer table:



8 ♦ Select the first field (Address), hold down the **SHIFT** key, and click on the last field (Terms).

9 ♦ With all of the fields selected, choose Add to move the fields to the Selected Fields selection list. Then choose OK.

The Column Editor dialog box opens again and the selected fields appear in the Fields in SmartDataObject selection list. The letter “u” now appears before each field name to indicate that the field is updateable:



NOTE: Usually, fields that define the unique key, in this instance CustNum, do not need to be updated.

10 ♦ Choose OK in the Column Editor dialog box to open Page 3 of the SmartDataObject Wizard again.

The selected fields now appear in the Fields Available to Visualization Objects selection list.

11 ♦ Choose Next in Page 3 of the SmartDataObject Wizard to open Page 4. Page 4 notifies you that you have successfully created the SDO:



4.5.3 Saving the New SmartDataObject

After creating the new SDO, you should save it using the filename `dcust`.

Follow these steps to save the SDO as `dcust`:

1 ♦ In Page 4 of the SmartDataObject Wizard, choose Finish.

A SmartDataObject window that represents the new SDO opens:



2 ♦ Choose the Save icon from the AppBuilder toolbar to open the Save As dialog box.

3 ♦ In the Save In combo box, select the working directory you created for this tutorial. The default working directory is `\Progress\wrk`.

4 ♦ In the File Name field, type **dcust.w** and choose Save.

The AppBuilder saves the new SDO and compiles it after saving the source file. The save produces the following five files:

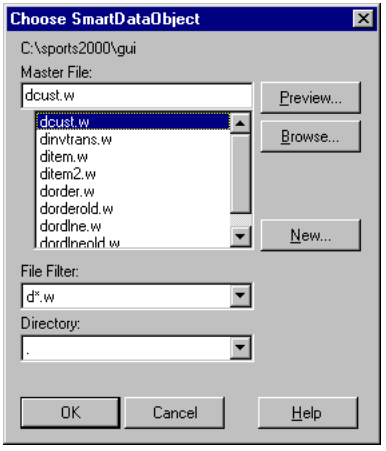
- `dcust.w` — Editable source file
- `dcust.i` — Include file with a list of the fields
- `dcust_cl.w` — Client proxy version of `dcust.w` for use with distributed applications
- `dcust.r` — Compiled version of `dcust.w`
- `dcust_cl.r` — Compiled version of `dcust_cl.w` for use with distributed applications

5 ♦ Click anywhere inside the SmartDataObject design window to select it and then choose File→ Close from the AppBuilder menu.

The new SDO should now through the Object Palette. Before you continue, verify that the SDO is available through the Object Palette.

6 ♦ Choose the SDO button in the Object Palette.

The Choose SmartDataObject dialog box opens. It shows the newly saved file in your working directory (indicated by the period (.) under Directory):



In the next section, you create a SDV that can display the fields queried by your new SDO.

4.6 Building SmartDataViewers

In this section, you use the SmartDataViewer Wizard to create the SDV, `vcust2`, one of the SDVs used in the Order Tracking System. The SmartDataViewer Wizard guides you through the SmartObjects creation by automatically launching the required dialog boxes in the correct sequence.

The SDV, `vcust2`, in the Order Tracking System illustrates the ease with which a SmartObject master can be reused in a single application or multiple applications. For example, there are two instances of `vcust2` in the Order Tracking System, and although the two instances are based on the same SmartObject master, they function differently.

The Find Customer window, `wfndcust`, is a child of the main window of the Order Tracking System. The Find Customer window includes a child window, the Display Customer window, `wdspcust`. In the Display Customer window, the SDV, `vcust2`, receives a customer record from the SDO, `dcust`, and displays certain fields from that record.

The Maintain Customer window, `wmntcust`, is a child of the main window of the Order Tracking System. In the Maintain Customer window, the instance of the SDV receives the same customer record from the SDO, `dcust`, as the instance of the SDV in the Find Customer window. However, instead of just displaying the fields, this instance of the SDV accepts changes to some of the displayed fields.

The following short procedure outlines the steps in building the SDV:

1 ♦ Start the SmartDataViewer Wizard.

To review this procedure, see the "Starting the SmartDataViewer Wizard" section.

2 ♦ Select a SDO.

To review this procedure, see Step 4 in the "Starting the SmartDataViewer Wizard" section.

3 ♦ Select the fields to display from the SDO.

To review this procedure, see the "Selecting the Fields to Display" section.

4 ♦ Arrange the fields in the design window.

To review this procedure, see the "Arranging Fields In the SmartDataViewer" section.

5 ♦ Add a SmartDataField to the Comments field.

To review this procedure, see Step 3 in the "Arranging Fields In the SmartDataViewer" section.

6 ♦ Modify field labels.

To review this procedure, see the "Adding and Modifying Field Labels" section.

7 ♦ Disable fields that should not be updatable.

To review this procedure, see the "Disabling Fields" section.

8 ♦ Save the new SDV.

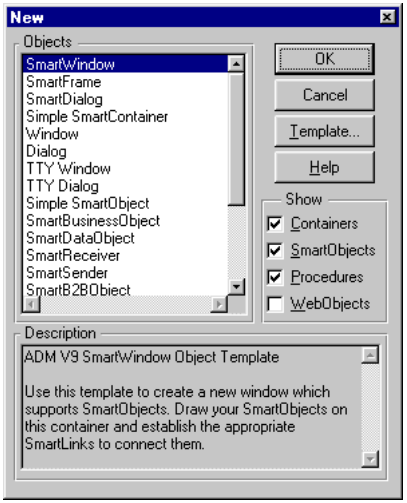
To review this procedure, see the "Saving the New SmartDataViewer" section.

4.6.1 Starting the SmartDataViewer Wizard

In this section, you start the SmartDataViewer wizard to help you create a SmartDataViewer. The wizard guides you through a series of dialog boxes that help you create the SmartDataViewer.

Follow these steps to start the SmartDataViewer Wizard:

1 ♦ Choose New on the AppBuilder toolbar to open the New dialog box:

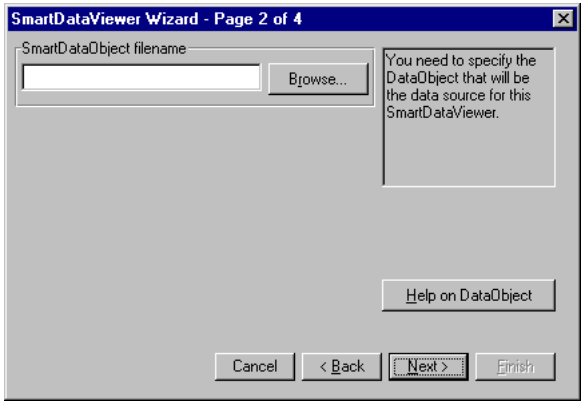


2 ♦ In the New dialog box, select SmartDataViewer and choose OK. Page 1 of the SmartDataViewer Wizard opens:

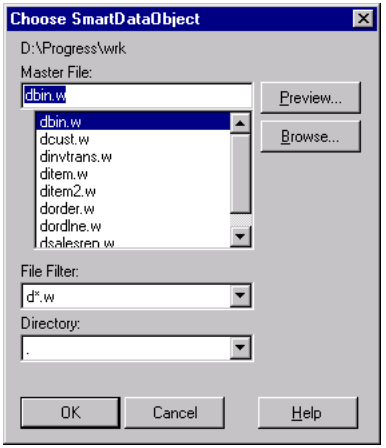


Like the SmartDataObject Wizard, the SmartDataViewer Wizard contains a Back button as well as a Next button. At any time prior to exiting the Wizard, you can return to a previous pages to modify your choices.

3 ♦ In Page 1 of the SmartDataViewer Wizard, choose Next to open Page 2:

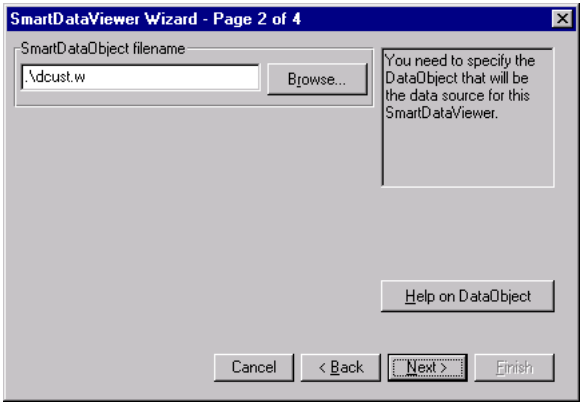


4 ♦ Choose Browse to open the Choose SmartDataObject dialog box:

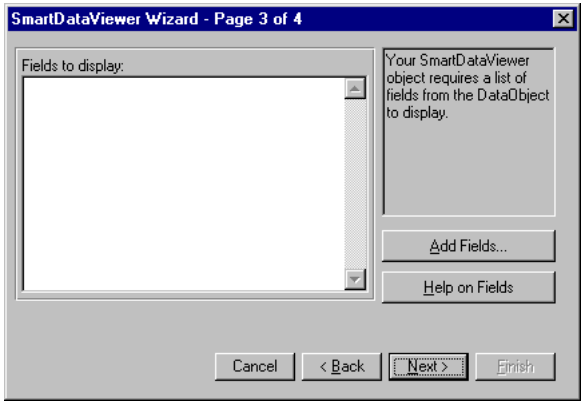


5 ♦ Select `dcust.w` and choose OK.

Page 2 of the SmartDataViewer Wizard opens again and `dcust.w` appears in the Select SmartDataObject filename field:



6 ♦ Choose Next to open Page 3 of the SmartDataViewer Wizard:



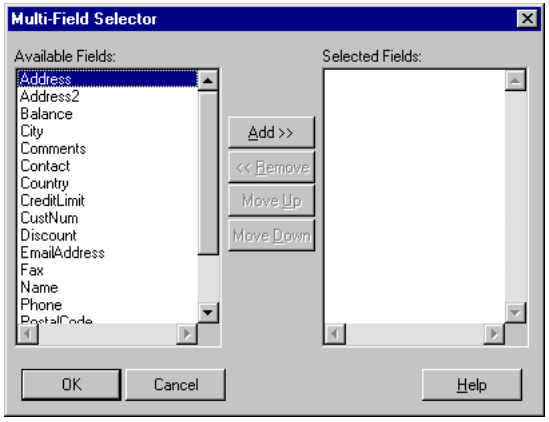
4.6.2 Selecting the Fields to Display

In this section, you select the fields you want displayed in the SDV.

Follow these steps to select the fields to display in the SDV:

1 ♦ In Page 3 of the SmartDataViewer Wizard, choose Add Fields to open the Multi-Field Selector dialog box.

The fields of the Customer table appear in the Available Fields selection list:



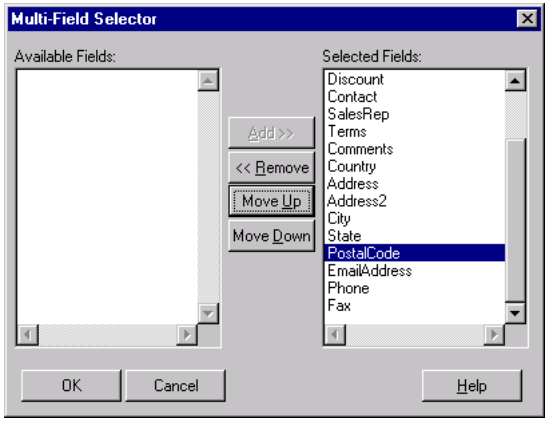
2 ♦ In the Available Fields list, select the first field, hold down the **SHIFT** key, and then select the last field.

3 ♦ With all of the fields selected, choose Add to move the selected fields to the Selected Fields list.

4 ♦ Click in the Selected Fields selection list and use the Move Up and Move Down buttons to arrange the selected fields in the following order:

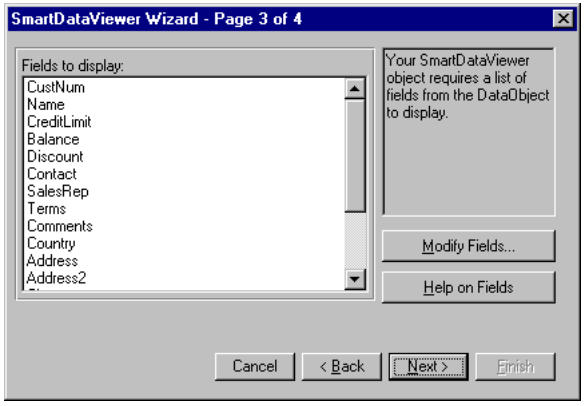
- CustNum
- Name
- CreditLimit
- Balance
- Discount
- Contact
- SalesRep
- Terms
- Comments
- Country
- Address
- Address2
- City
- State
- PostalCode
- EmailAddress
- Phone
- Fax

The Selected Fields selection list in the Multi-Field Selector dialog box should look like the following figure:



5 ♦ Choose OK to open Page 3 of the SmartDataViewer Wizard again.

The fields you selected and arranged in the Multi-Field Selector dialog box now appear in the Fields to display text box:



The button labeled Add Fields has changed to Modify Fields. You can use Modify Fields to return to the Multi-Field Selector and rearrange, add, and delete fields.

6 ♦ In Page 3 of the SmartDataViewer Wizard, choose Next to open Page 4 of the SmartDataViewer Wizard.

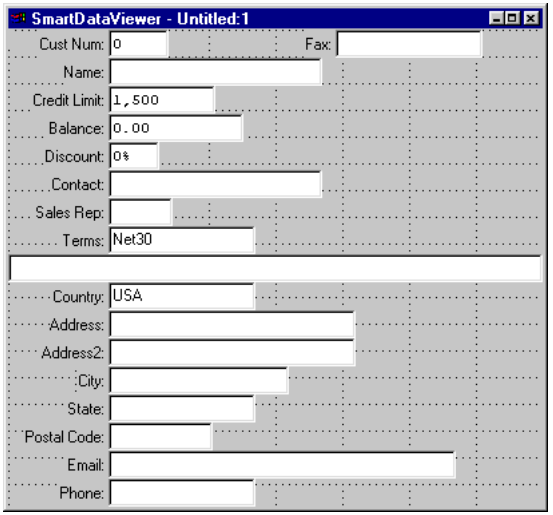
This page notifies you that you have successfully created the SmartDataViewer:



7 ♦ Choose Finish.

NOTE: You might receive a message that the comments field is too large and its label will be omitted. You can choose OK to dismiss this message, as you will adjust the display and add a label later.

The SmartDataViewer window opens:



In the next section, you are going to make changes to the fields in the SDV.

4.6.3 Arranging Fields In the SmartDataViewer

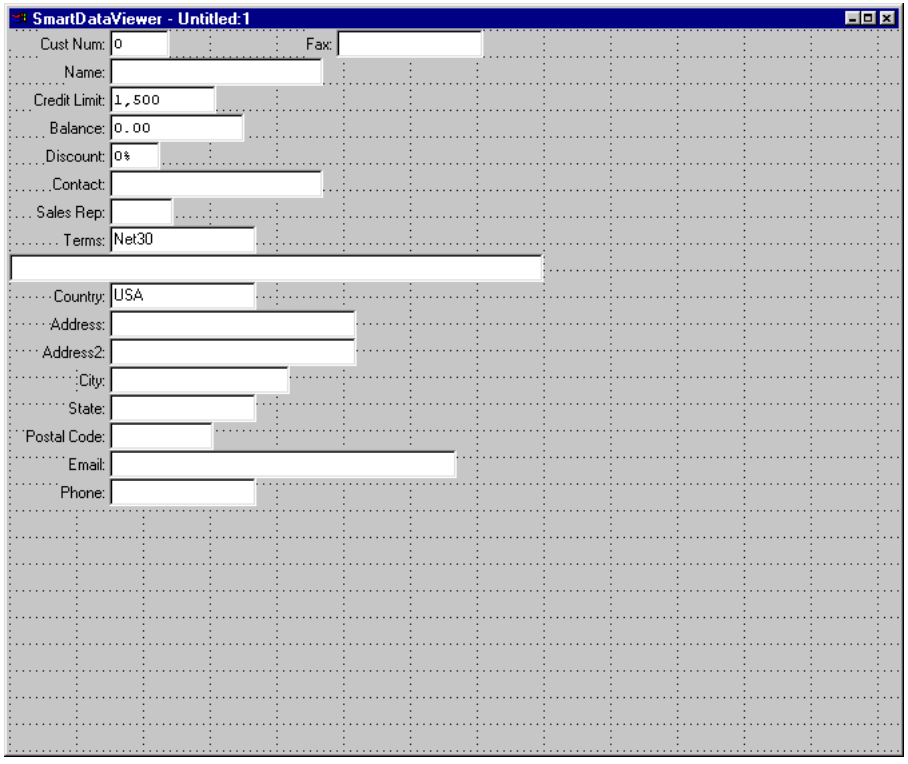
In this section, you resize the window, change the layout of the fields in the SDV, and attach an editor SmartDataField to the Comments field.

Follow these steps to change the layout of the fields:

1 ♦ Drag the bottom-right corner of the SDV down to the right to proportionally resize the viewer to about twice the width and height of the original window.

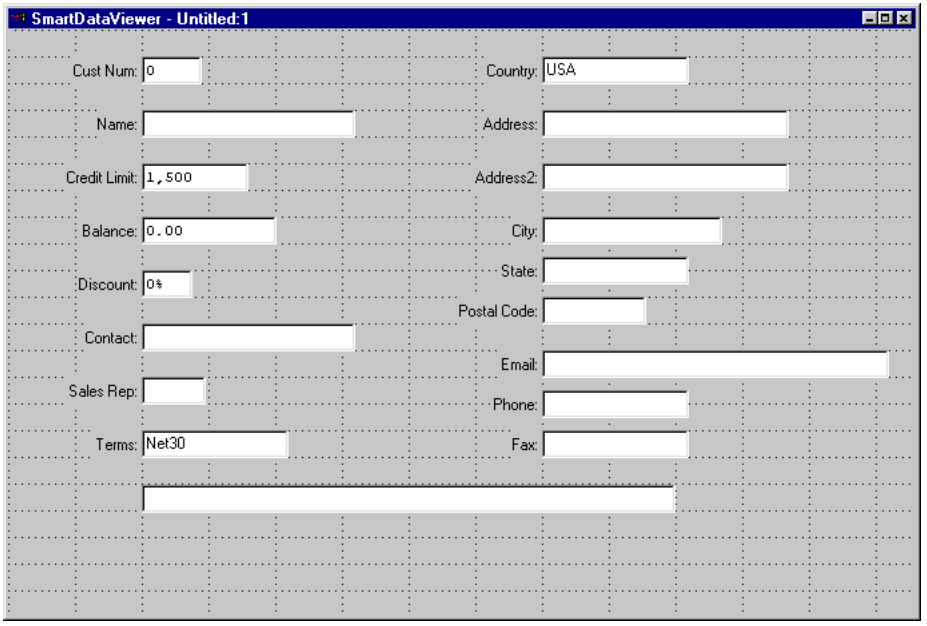
NOTE: The frame automatically resizes to fit the new dimensions of the window.

When you finish resizing the design window, it should look like the following figure:



You can adjust the size of the window more precisely later, if you need to.

2 ♦ Drag the fields into the positions shown below. Leave space around the edges of the design window and between the columns of fields for text labels:



NOTE: The CustNum, CreditLimit, Balance, Discount, Terms, and Country fields appear with initial or default values.

3 ♦ Select the Text icon from the Object Palette and click to the left of the unlabeled Comments field to add a label for it.

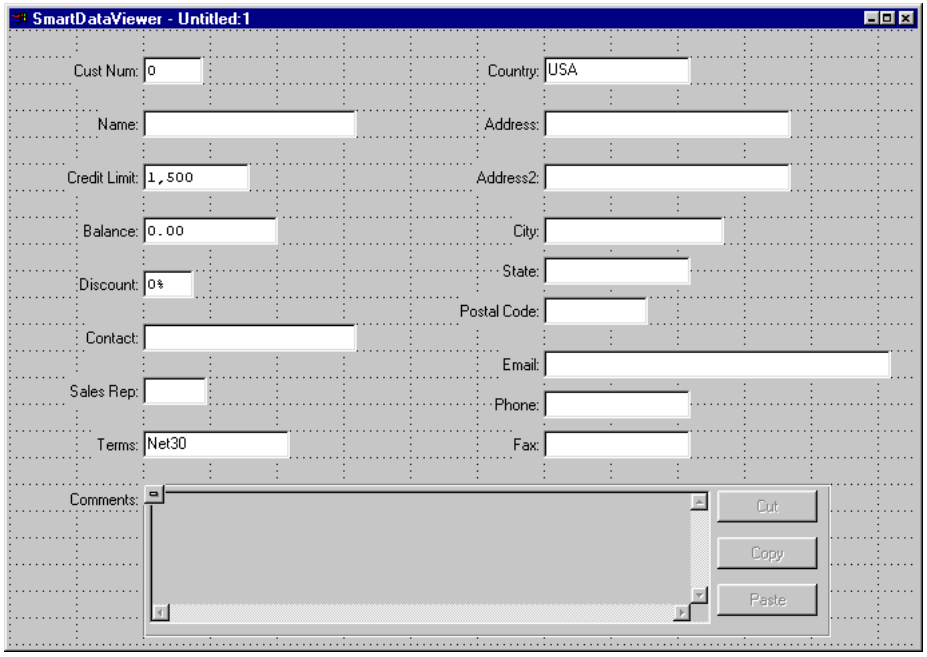
4 ♦ Double click on the text box to open the TEXT property sheet, type **Comments:** into the Text field, and then choose OK.

5 ♦ Size the text box on the SDV window to display the full label.

6 ♦ Select the SmartDataField icon  on the Object Palette to open the Choose SmartDataField dialog box.

7 ♦ Select `stxtfld.w` from the selection list and choose OK.

8 ♦ Click inside the Comments field to place the SmartDataField into the Comments field. The window now looks similar to the one in the following figure:



4.6.4 Adding and Modifying Field Labels

In this section, you remove the label from the Address2 field and change the label of the Postal Code field to a shorter label.

Follow these steps to modify the labels:

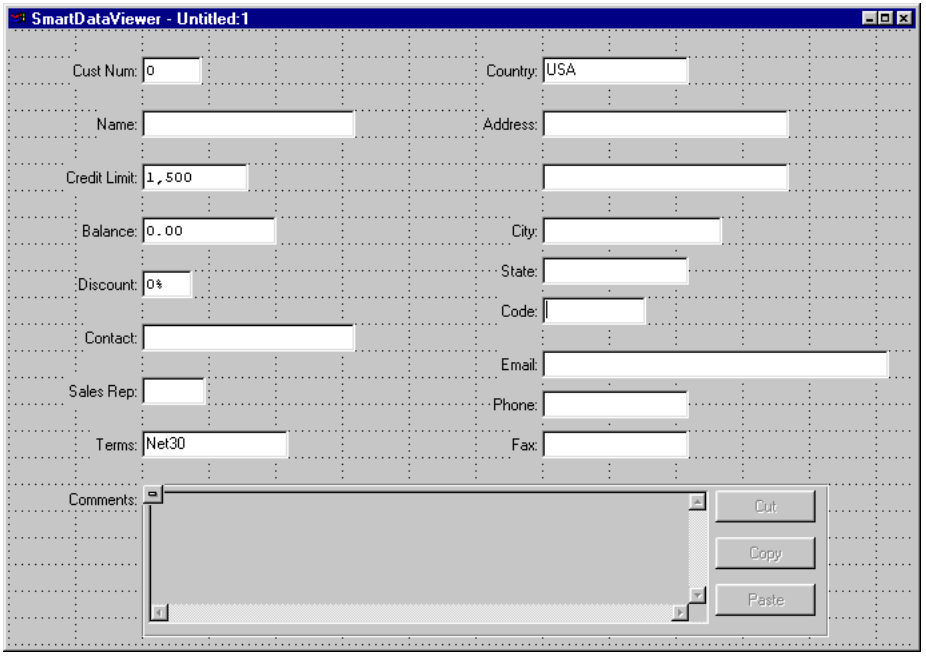
1 ♦ Double-click the Address2 field to open its property sheet, select No Label to suppress displaying a label for this field, and choose OK to save the change.

2 ♦ Click the Postal Code field to select it. The AppBuilder's Current Object field and Title field display the Postal Code field and its label.

3 ♦ Type Code in the Label field of the AppBuilder main window.

4 ♦ Click anywhere in the SmartDataViewer window. The “Postal Code” label changes to “Code”.

After making these label changes, the SmartDataViewer window looks like the following figure:

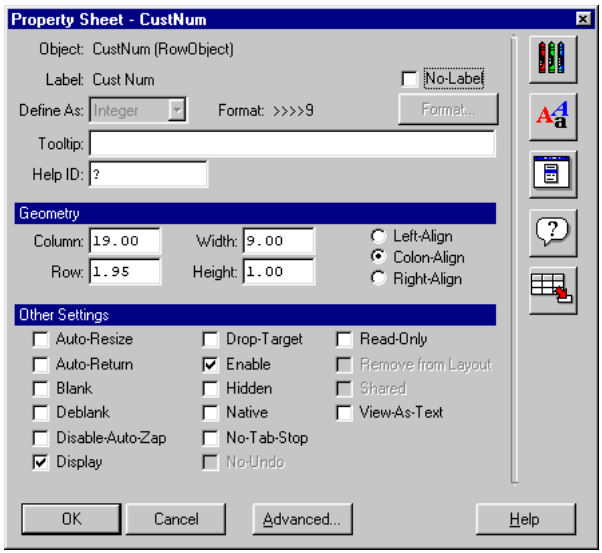


4.6.5 Disabling Fields

In this section, you disable the CustNum, Name, CreditLimit, Balance, Discount, SalesRep, and Terms fields to prevent users from making changes to these fields. Disabling these fields changes the role of `vcust2` in the Customer Update window. Before disabling these fields, users could view and make changes to them. After you disable these fields, this instance of `vcust2` limits its use to small changes in the customer’s address, telephone, contact, e-mail, and fax information.

Follow these steps to disable the fields:

1 ♦ Double-click the CustNum field to open its property sheet:



2 ♦ In the Other Settings section, choose the Enable toggle box to deselect it.

3 ♦ Choose OK to return to the design window.

4 ♦ Repeat Step 1 through Step 3 to disable the Name, CreditLimit, Balance, Discount, SalesRep, and Terms fields.

4.6.6 Saving the New SmartDataViewer

After creating the SDV and making the changes to the fields, you should save the new SDV. Because you built this SDV from the SDO, `dcust.w`, its default name would be `vcust.w`. However, you are going to save the new viewer using a different name.

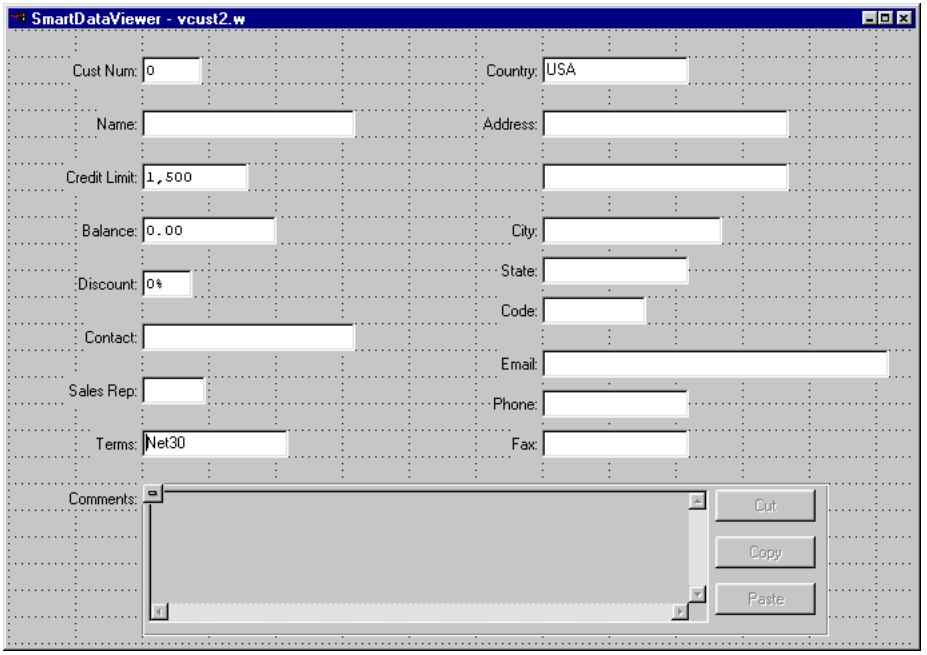
Follow these steps to save the new SDV:

1 ♦ Choose Save on the AppBuilder main window to open the Save As dialog box.

2 ♦ In the Save In combo box, select the working directory you created for this tutorial. The default directory is `\Progress\wrk`.

3 ♦ In the File name text box, type **vcust2.w** and choose Save.

The AppBuilder saves the new SDV and displays its filename in the title bar of the SmartDataViewer window:



4 ♦ Click anywhere inside the SmartDataViewer window and choose **File**→ **Close** from the AppBuilder menu to close the window.

You now have a SDV and SDO. In the next section, you fabricate a SDB.

4.7 Building SmartDataBrowsers

In this section, you use the SmartDataBrowser Wizard to create the SDB, `bordlne`, which is similar to the SDB, `bordlne2`, in the Order Tracking System. Like the SmartDataObject Wizard and the SmartDataViewer Wizard, the SmartDataBrowser Wizard guides you through the steps necessary to create a SDB, giving you information about each step in the process. The SmartDataBrowser Wizard also allows you, at any point, to go back and modify the work you did in previous steps.

In the main window of the SportsPro Order Tracking System, `bordlne2` identifies and retrieves an Order Line record based on end-user input, and sends the record to other SmartObjects that require Order Line information to perform their functions.

The following short procedure outlines the steps in building the SDB:

1 ♦ Start the SmartDataBrowser Wizard.

To review this procedure, see the "Starting the SmartDataBrowser Wizard" section.

2 ♦ Select a SDO.

To review this procedure, see Step 4 in the "Starting the SmartDataBrowser Wizard" section.

3 ♦ Select the fields to display from the SDO.

To review this procedure, see the "Selecting Fields for the SmartDataBrowser" section.

4 ♦ Resize the SDB.

To review this procedure, see the "Resizing the SmartDataBrowser" section.

5 ♦ Save the new SDB.

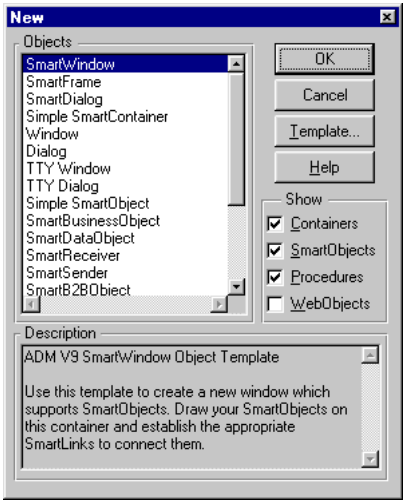
To review this procedure, see the "Saving the New SmartDataBrowser" section.

4.7.1 Starting the SmartDataBrowser Wizard

In this section, you start the SmartDataBrowser wizard to help you create a SmartDataBrowser. The wizard guides you through a series of dialog boxes that help you create the SmartDataBrowser.

Follow these steps to start the SmartDataBrowser Wizard:

1 ♦ Choose New in the AppBuilder toolbar to open the New dialog box:

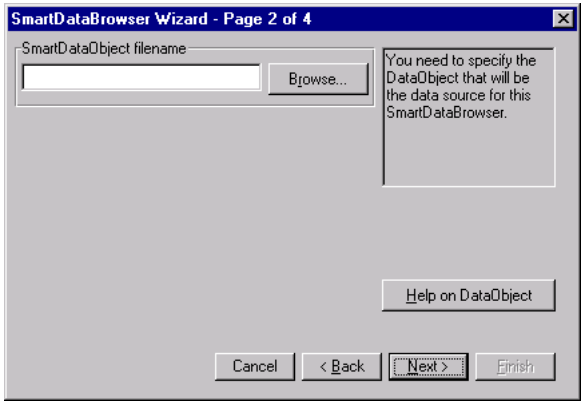


2 ♦ In the New dialog box, select SmartDataBrowser and choose OK to open Page 1 of the SmartDataBrowser Wizard:



Like the SmartDataObject Wizard, the SmartDataBrowser Wizard contains a Back button as well as a Next button. At any time prior to exiting the Wizard, you can return to previous pages to modify your choices.

3 ♦ In Page 1 of the SmartDataBrowser Wizard, choose Next to open Page 2:



4 ♦ Choose the Browse button to open the Choose SmartDataObject dialog box.

5 ♦ Select `dordlne.w` and choose OK to return to Page 2 of the SmartDataBrowser Wizard. The file `dordlne.w` now appears in the SmartDataObject filename field:



Selecting a SDO gives you a data source for the SDB. In the next section, you select the fields that you want to appear in the SDB.

4.7.2 Selecting Fields for the SmartDataBrowser

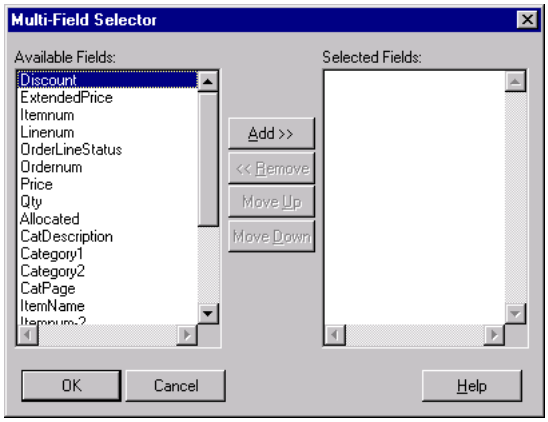
Now that you have a data source for the SDB, you can select the fields to display in the SDB.

Follow these steps to specify the data source and select the fields:

1 ♦ In Page 2 of the SmartDataBrowser Wizard, choose Next to open Page 3:



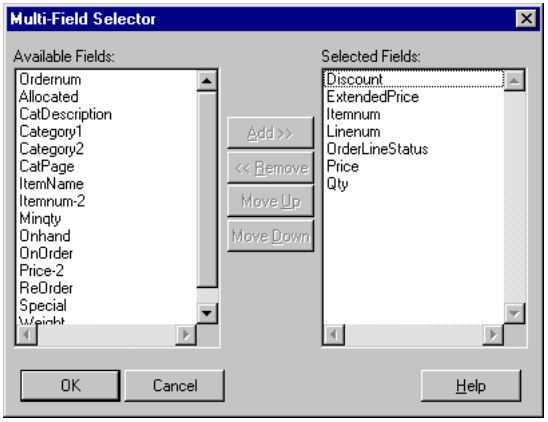
2 ♦ In Page 3 of the SmartDataBrowser Wizard, choose Add Fields to open the Multi-Field Selector dialog box. The fields in the Order Line table appear in the Available Fields selection list:



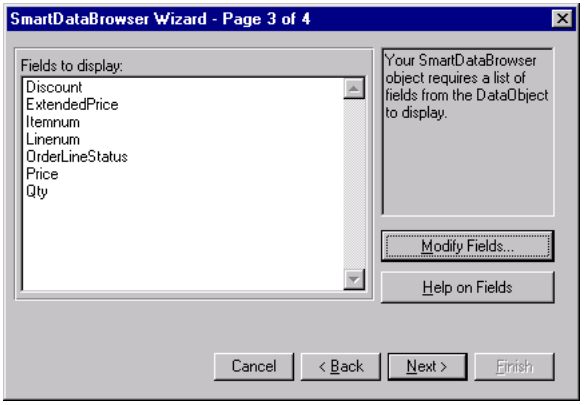
3 ♦ Use the **CTRL** key to select the following fields and choose Add to move the fields to the Selected Fields list:

- Discount
- ExtendedPrice
- Itemnum
- Linenum
- OrderLineStatus
- Price
- Qty

When you finish selecting the fields, the Multi-Field Selector dialog box should look like the one in the following figure:

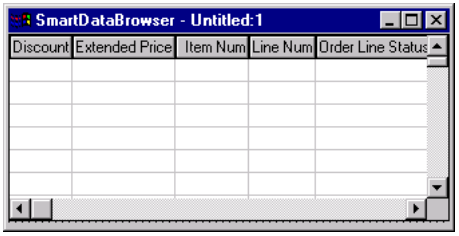


4 ♦ Choose OK to return to Page 3 of the SmartDataBrowser Wizard. The fields you selected appear in Fields to display selection list:



5 ♦ In Page 3, choose Next to open Page 4 of the SmartDataBrowser Wizard. Page 4 notifies you that you have successfully created the SDB.

6 ♦ Choose Finish. The SDB opens:



You now have a fully functional SDB that you can use in an application. In the next section, you enhance the functionality and appearance of the SDB by resizing it.

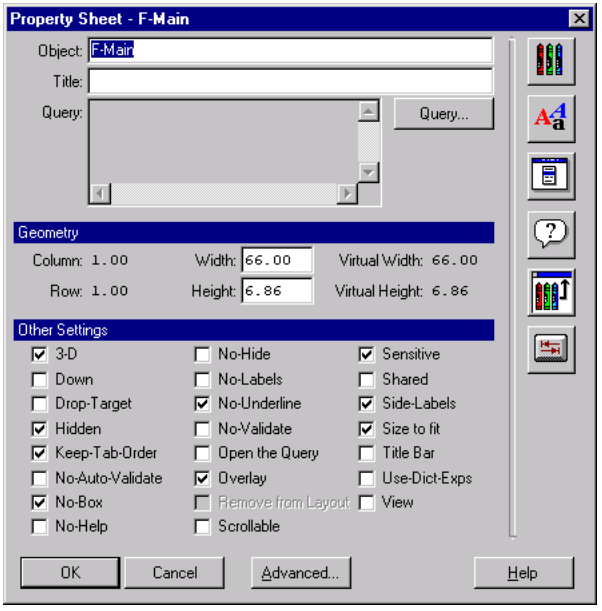
4.7.3 Resizing the SmartDataBrowser

To improve the usaability of the SDB, you should resize the frame and the browse object to make the SDB wide enough to display all of its fields without requiring horizontal scrolling.

Follow these steps to resize the frame:

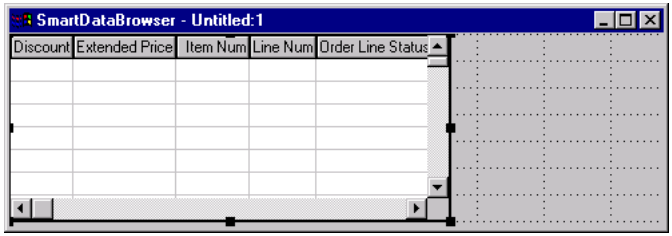
1 ♦ Click the outside edge of the frame. The Object field in the AppBuilder's main window identifies the selected object as F-Main.

2 ♦ Choose the Object Properties button in the AppBuilder toolbar to open the property sheet for the frame:



3 ♦ In the property sheet, type **98.00** for Width and **6.91** for Height. These dimensions enable the SDB to display all of its fields and several records at once.

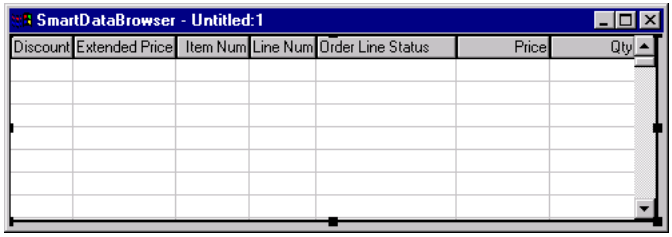
4 ♦ Choose OK. The design window resizes to the new dimensions of the frame:



Now you need to resize the browse object.

5 ♦ Click once anywhere within the white space of the browse object to select it. Select the browse object's right resize handle and drag the edge of the browse object until it fills the design window.

6 ♦ Drag the column dividers to size the columns appropriately for the data displayed in each column:



4.7.4 Saving the New SmartDataBrowser

Now that you have completed the new SDB, you should save it.

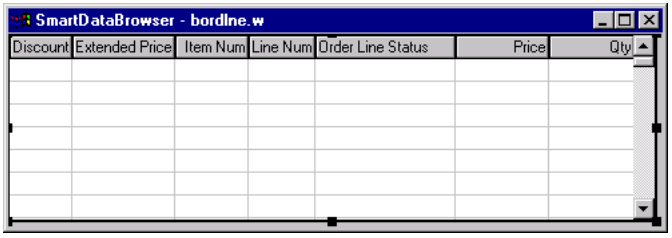
Follow these steps to save the new SDB:

1 ♦ Choose File→ Save from the AppBuilder menu to open the Save As dialog box.

2 ♦ In the Save in combo box, select your working directory. The default working directory is `\Progress\wrk`.

3 ♦ In the File name text box, type `bordlne.w` and then choose Save.

The AppBuilder saves the new SDB and displays its filename in the title bar of the window:



4 ♦ Close the SmartDataBrowser design window by choosing the Close button in the upper-right corner (or pressing **F8**).

You have now completed fabricating a number of SmartObjects. If you would like to test your knowledge of SmartObjects and the AppBuilder, you can take what you have learned and create the Maintain Customer window, `wmntncust`, using the SmartObjects you created. To create this dialog box, you need to create a new SmartWindow, add the graphic banner, and add your new SmartObjects. You can also add stripes and an update SmartPanel, `pupdsav` (in `\src\adm2`). You can open `wmntncust`, the Maintain Customer window, to see how it is assembled and what SmartLinks you need to add.

5

Where To Go From Here

This book provides you with a practical introduction to the Progress Application Development Environment (ADE), SmartObject technology, and the basic techniques of AppBuilder application assembly and SmartObject fabrication.

You now have the information you need to create effective, easily maintainable end-user applications that require little or no manually written 4GL code. However, the story does not end here. The Progress Application Development Environment is an open system, many features of which are fully customizable.

For example, you are not limited to the types of SmartObject templates that are represented in the AppBuilder's Object Palette. While this starter set of templates is designed to address a wide range of database-oriented application development needs, you might want to create your own SmartObject templates based on the business standards and practices of individual enterprises. The ADE gives you the full power to do this and to create your own Cue Cards and Wizards to accompany your custom SmartObject templates.

In addition, the Object Palette itself is customizable, allowing you to add buttons, directory lists, and file filters for the custom SmartObject templates you create (and for the SmartObject masters that are based on your custom templates).

Another aspect of the ADE's openness and flexibility is its full access to the depth and power of the Progress 4GL and its seamless integration with the Progress RDBMS and DataServer Architecture and with the open AppServer in an n-tier environment.

Progress Software Corporation is fully committed to helping all of its customers make a smooth transition to the new way of building new applications provided by the ADE. This chapter describes the resources available to help you expand and deepen your knowledge, and to gain ever greater control over the power of Progress.

This chapter consists of the following sections:

- The Progress Documentation Set
- Education Services
- Hands-on Training
- Consulting Services
- Technical Services

5.1 The Progress Documentation Set

The Progress documentation set provides detailed information on the functionality of all Progress products, providing much practical guidance on programming and data administration techniques in addition to thorough reference materials.

Of greatest relevance to deepening your knowledge of the AppBuilder and SmartObject technology are the *Progress AppBuilder Developer's Guide*, the *Progress ADM 2 Guide*, and the *Progress ADM 2 Reference*. These three manuals expand on each of the topics introduced in this book, include detailed information on the many ways in which you can customize and extend the AppBuilder, and provide reference material detailing the method and event procedures that constitute the common currency of SmartObject functionality and interoperability. For a brief description of each book in the documentation set, see the "Preface" of this book.

5.2 Education Services

Progress Software's comprehensive training classes help you keep pace with today's rapidly changing technology. By offering a thorough understanding of the rich functionality of our products, Progress Education Services can help your developers reduce the overall development cycle and ultimately reduce the time-to-market or time-to-production of your applications, thereby providing your developers with the skills to stay ahead of your competition

Progress Education Services offers a selection of courses taught by experts in state-of-the-art facilities located throughout the United States and Canada. On-site training is also available for the convenience of our customers who want to take our interactive, hands-on courses directly at their workplace. Regardless of the location, Progress offers a wide range of pricing programs tailored to meet any budget.

By enrolling in a Progress course, you receive an education from a professional who has made software training a career focus. Progress instructors are specialists who receive technical training in various aspects of Progress products. A Progress education provides you with:

- High-quality course materials and classes taught using the latest available technology
- Experienced instructors with access to Progress benchmarking and source materials
- Hands-on courses developed in conjunction with Progress software engineers

5.3 Hands-on Training

Beginning with the Version 8.2 curricula, all Progress courses adopted a task-based orientation that stresses hands-on activities. The goal of these courses is to teach students how to do their jobs rather than focus on the features and benefits of Progress products. Courses emphasize need-to-know information rather than nice-to-know information, resulting in a concentrated but highly informative class.

Course materials are structured around hands-on activities that offer students the opportunity to build, whenever possible, a substantive application of their own while working on course labs. The Progress course book, organized by tasks, later serves as a handy, on-the-job reference tool for students.

For more information on education pricing programs, or on available courses, call the Education Sales Office at 800-477-6473, extension 4217.

For a complete list of courses, and to register for classes online, visit the Progress Web site at `www.progress.com/education`. To speak with a Training Registrar, call 800-477-6473, extension 4452, or fax your request to: 781-280-4432.

5.4 Consulting Services

The goal of Progress Consulting Services is to provide business solutions that enable you to fully leverage the unique capabilities of your Progress-based application and maximize your investment in a Progress solution.

Progress Consulting offers a range of services for all lifecycle phases of your Progress-based application—from analysis and design, through development and deployment to system maintenance. The selection of services includes, but is not limited to:

- Developing E-business Environments
- Web-Enabling Applications
- Custom Development Solutions
- Packaged Software Implementation
- Migration/Conversion Services
- Database Design
- Technical Services
- Project Management

Consulting services are offered in the form of either short-term engagements (HeadStart Services) or strategic alliances (SMART Family of Services) or a combination of the two. Progress' HeadStart Services provide the expertise of a qualified consultant on a short-term basis (usually two to five days) to assist companies with the development and deployment of a Progress solution. The SMART Family of Services are engagements that are customized to address a company's unique business requirements.

The Progress Consulting Team includes some of the most knowledgeable software professionals in the industry. With backgrounds in systems analysis, customer support, and software training, Progress consultants are experts in project management, application design, and product implementation.

Progress Consulting Services can direct and support you during all phases of large-scale projects. This allows you to get the expertise you need on a project-by-project basis—long-term or short-term—without increasing your in-house staff. As part of your development team or as individual contributors, Progress' professional consultants bring a wealth of knowledge and expertise to your organization.

To add an experienced, creative force to your organization—either a team or an individual who can take you beyond the ordinary—call 781-280-4290. For more information on our selection of services, visit our Web site at `www.progress.com/consulting`.

5.4.1 Phone-in Consultation

Consulting Services also offers phone-in consulting to handle your small how-to questions. This service does not replace traditional on-site services but addresses, early in the process, conceptual and direction issues you might have. Call 781-280-3001 for a description of the charges associated with phone-in consultation.

5.5 Technical Services

Progress Software maintains a long-term commitment to superior product performance and productivity. The Progress ADE was designed to be as intuitive and easy to use as possible. Nonetheless, it is sometimes necessary to seek technical assistance.

For those customers who have purchased maintenance contracts, our dedicated and knowledgeable Technical Services professionals are ready to assist you with your issues when commercial products fail to work as documented.

5.5.1 Contacting Technical Services

Technical Support provides assistance with products according to the Progress Software Product Life Cycle for Progress Products Document.

Basic Service is provided for Active and Functionally Stable versions of the product. Limited Basic Service is offered for products and versions that have reached the Mature or Retired status.

Our goal is to provide the best support possible to our customers. We define a customer as an organization that has a valid maintenance contract with Progress Software Corporation. To maintain high quality support, it is important that we service and support those customers who have contracts with Progress Software.

5.5.2 Progress Technical Support Centers

The Americas Support Center supports Progress Software distributors, ISVs, and direct end users in North and South America. Support is provided in English for North America and Spanish and Portuguese for Latin America through phones, faxes, e-mail messages, and the Web.

Technical Support Center In the Americas

The Technical Support Center in the Americas is divided into five groups:

- Language and Tools supports the application development process.
- Database Management supports the Progress database, networking, and deployment issues.
- Dataserver supports WebSpeed, Progress/400, and non-Progress Database configurations.
- Apptivity supports Apptivity-related issues.
- Latin America supports issues from the Latin America region.

Technical Support Center In Europe, Middle East, and Africa

The Europe, Middle East, and Africa (EMEA) Technical Support Center supports Progress Software distributors, ISVs, and direct end users in the EMEA region. Support is provided in 10 different languages to satisfy the requirement that EMEA supply telephone coverage in native languages. The engineers all belong to a language team to ensure proper telephone coverage. They also belong to one or more skill groups, which are defined around different areas of the Progress product set.

Technical Support Center In Asia Pacific

The Asia Pacific Support Center supports Progress Software subsidiaries, distributors, ISVs, and direct end users in the Asia Pacific region. Support is provided in English through phones, faxes, e-mail messages, and the Web.

For further information, please refer to the document *How to Use Technical Support* located on the Progress Software Web site at `http://www.progress.com/support`.

5.5.3 TechWeb

As a direct Progress Software customer with maintenance, you have 24-hour access to the Technical Support call tracking system. This online resource enables you to submit and track your product issues.

5.5.4 Progress Knowledge Base

The Progress Knowledge Base is an up-to-date technical reference database that provides solutions to known Progress problems, explanations of error messages, and other relevant technical information. You can access the Progress Knowledge Base through the Internet at `http://www.progress.com/support/kb`.

Index

A

ActiveX (OCX) controls 3–50
  adding spin boxes 3–51

ActiveX Automation objects
  defined 3–50

Actuate e.Report Designer Professional
  defined 1–5

ADE <italics>See<Default Paragraph Font> Application Development Environment 1–2

Advanced Editing Features
  Alias expansion 1–18
  Color coding 1–17
  Embedded color coding 1–17
  SmartPaste 1–18
  Syntax expansion 1–17
  Syntax indenting 1–18

Advanced Properties dialog box 3–37

Alias expansion
  advanced editing features 1–18

AppBuilder
  current object fields 1–10
  customizing Object Palette 1–16
  defined 1–4
  File field 1–10
  Label field 1–10
  main window 1–7
  Master field 1–10
  Menu bar 1–7
  Object field 1–10
  Object Palette 1–11
  OCX (ActiveX) field 1–10
  overview 1–6
  property sheet 3–8
  Section Editor defined 1–4
  starting 1–6
  status area 1–11
  Text field 1–10
  Title field 1–10

AppBuilder filename conventions 4–9

AppBuilder main window
  elements 1–7
  Window Properties Window option 3–53

AppBuilder menu bar
  Compile menu 1–7
  Edit menu 1–7
  File menu 1–7
  Help menu 1–8
  Layout menu 1–8
  Options menu 1–7
  Tools menu 1–7
  Window menu 1–8

AppBuilder status area
File status 1–11
Lock status 1–11
Object Palette status 1–11
Page status 1–11, 3–77

AppBuilder toolbar
Colors button 1–10
Development Mode button 1–10
Edit code button 1–9
List Objects button 1–9
New button 1–8
Object Properties button 1–9
Open button 1–8
Print button 1–9
Procedure Properties button 1–9
Run button 1–9
Save button 1–8
Stop button 1–9

Application Compiler
defined 1–5

Application Debugger
defined 1–5

Application Development Environment (ADE)
desktop 1–2
starting 1–2
toolset 1–3

Audience 1–ix

B

Bold typeface
as typographical convention 1–xi

Browse object
AppBuilder Object Palette 1–12

Browser tab
Find Order window 2–45
in SmartFolder 2–48

Business solutions 5–4

Button object
AppBuilder Object Palette 1–13

Buttons
adding help text 3–33
adding images 3–33
adding ToolTips 3–33

C

Call tracking system
24-hour access 5–7

Changing orders
Order Entry/Maintenance window 2–47

Choose Color dialog box 3–17

Choose Control dialog box 3–51

Choose Event dialog box 3–56

Choose Image Files dialog box 3–12

Choose SmartDataObject dialog box 4–20, 4–24

Choose SmartDataViewer dialog box 3–26

Color coding
advanced editing features 1–17

Colored text labels
adding 3–54

Colors button
AppBuilder toolbar 1–10

Column Editor dialog box 4–16

Combo-Box object
AppBuilder Object Palette 1–13

Compile menu
AppBuilder 1–7

Compiling
sample files 2–6

Connect Database dialog box 2–4

Context-sensitive help
accessing 1–5

Courses
complete list 5–4

Create Database dialog box 2–3

Creating applications
analysis and design 4–10
assembling 4–10
building 4–10

Creating SmartDataObjects 4–12

CSComboBox object
AppBuilder Object Palette 1–14

CSSpin object
AppBuilder Object Palette 1–13

Cue Cards 3–6

Current object field
File field 1–10
Label field 1–10
Master field 1–10
Object field 1–10
OCX (ActiveX) field 1–10
Text field 1–10
Title field 1–10

Current object fields
AppBuilder 1–10

CustNum field property sheet 4–33

D

Data Administration tool
defined 1–3

Data Dictionary
connecting to Sports2000 database 2–2
creating a connection 2–2
defined 1–3

Data Dictionary window 2–4

Database connection
Connect Database dialog box 2–4

DB-Fields object
AppBuilder Object Palette 1–12

Design Page dialog box 3–73

Design page number
changing 3–77

Designing an application 4–10

Desktop
menu bar 1–2
toolbar 1–2

Detail tab
in SmartFolder 2–48

Development Mode button
AppBuilder toolbar 1–10

Dialog boxes
Advanced Properties 3–8
Choose Color 3–17
Choose Control 3–51
Choose Event 3–56
Choose Image Files 3–12
Choose SmartDataObject 3–24, 4–20, 4–24
Choose SmartDataViewer 3–26
Column Editor 4–16, 4–18
Connect Database 2–4
Create Database 2–3
Design Page 3–73
Goto Page 3–77
Insert Procedure Call 3–58
List Objects 3–19, 3–35
Multi-Field Mapping 3–43
Multi-Field Selector 4–17, 4–25, 4–39
New 3–5, 4–13, 4–22, 4–36
Pages 3–73
Procedure Settings 3–72
Property Editor 3–53
SmartLinks 3–29
Startup on Page 3–74

Dictionary Startup dialog box 2–3

Display Customer window 2–36

Drag and Drop module 2–15
components 2–17

Dynamic browser
Find Order window 2–44

Dynamic Buffer/Temp Table module 2–12
components 2–14

Dynamic query
in Find Order window 2–45

Dynamic Query module 2–10
components 2–11

E

Edit code button
AppBuilder toolbar 1–9

Edit menu
AppBuilder 1–7

Editor object
AppBuilder Object Palette 1–13

Education pricing programs 5–4

Embedded color coding
advanced editing features 1–17

Error messages
displaying descriptions 1–xii

F

File field
AppBuilder 1–10

File menu
AppBuilder 1–7

File status
AppBuilder status area 1–11

File types
naming conventions 4–9

Fill-In object
AppBuilder Object Palette 1–13

Find Customer window 2–34

Find Order window 2–44, 2–45
Browser section 2–45
dynamic browser 2–44
order filter section 2–45
search query 2–44
Selected Fields Displayed 2–45
SmartFilter 2–44

Frame object
AppBuilder Object Palette 1–12

G

Goto Page dialog box 3–77

H

Help
Progress messages 1–xii

Help file
adding 3–35

Help menu
AppBuilder 1–8

Help text
adding to buttons 3–33

I

Image object
AppBuilder Object Palette 1–13

Images
adding to buttons 3–33
adding to SmartWindow 3–11
property sheet 3–12

Information and support
additional sources 5–2

Insert Procedure Call dialog box 3–58

Instance attributes
  SmartFolder 3–69

Inventory Trans dialog box 2–50

Italic typeface
  as typographical convention 1–xi

Item Maintenance window 2–49

Item Selection window 2–41
  SmartSelect Browser 2–41

K

Keystrokes 1–xi

L

Label field
  AppBuilder 1–10

Layout menu
  AppBuilder 1–8

List Objects button
  AppBuilder toolbar 1–9

List Objects dialog box 3–19

Lock status
  AppBuilder status area 1–11

Logical pages
  adding to SmartFolders 3–72
  Page 0 3–73
  SmartDataViewer 2–47

M

Main window
  Order Line section 2–47

Main window frame
  property sheet 4–42

Maintain Customer window 2–37
  SmartDataField 2–37
  SmartToolBar 2–37

Maintenance contract 5–6

Manual, organization of 1–ix

Master field
  AppBuilder 1–10

Menu bar
  AppBuilder 1–7
  Desktop 1–2

Messages
  displaying descriptions 1–xii

Modifying OCX spin boxes 3–53

Monospaced typeface
  as typographical convention 1–xi

Multi-Field Mapping dialog box 3–43

Multi-Field Selector dialog box 4–17, 4–25, 4–39

N

Named-events module 2–24

Navigation SmartPanel
  adding 3–79
  Item Maintenance window 2–49
  Order section 2–40

New button
  AppBuilder toolbar 1–8

New dialog box 3–5, 4–13, 4–22, 4–36

O

Object Component Extension (OCX)
  defined 3–50

Object field
  AppBuilder 1–10

Object Palette
Button icon 3–30
controlling with mouse 1–16
customizing 1–16, 5–1
defined 1–11
OCX icon 3–51
Options 1–11
Rectangle icon 3–13
setting characteristics 1–11
SmartDataBrowser icon 3–75
SmartDataObject icon 3–24, 3–44
SmartDataViewer icon 3–26, 3–28, 3–77
SmartPanel icon 3–79
Text icon 3–16, 3–54

Object Palette objects
Browse 1–12
Button 1–13
Combo-Box 1–13
CSComboBox 1–14
CSSpin 1–13
DB-Fields 1–12
Editor 1–13
Fill-In 1–13
Frame 1–12
Image 1–13
OCX 1–13
Pointer 1–12
PSTimer 1–14
Query 1–12
Radio-Set 1–13
Rectangle 1–13
Selection-List 1–13
Slider 1–13
SmartB2BObject 1–15
SmartBusinessObject 1–15
SmartConsumer 1–15
SmartContainer 1–15
SmartDataBrowser 1–14
SmartDataField 1–14
SmartDataObject 1–14
SmartDataViewer 1–14
SmartFilter 1–14
SmartFolder 1–14
SmartObject 1–14
SmartPanel 1–14
SmartProducer 1–15
SmartReceiver 1–15
SmartRouter 1–15
SmartSender 1–15
SmartToolBar 1–14
Text 1–13
Toggle-Box 1–13

Object Palette status
AppBuilder status area 1–11

Object Properties button
AppBuilder toolbar 1–9
creating SmartWindows 3–7

OCX (ActiveX) field
AppBuilder 1–10

OCX controls
adding spin boxes 3–51
licensed by Progress 3–50
third-party objects 3–50

OCX object
AppBuilder Object Palette 1–13

Online Help
accessing context-sensitive help 1–5
AppBuilder 1–8
defined 1–5

On-site training 5–3

Open button
AppBuilder toolbar 1–8

Options menu
AppBuilder 1–7

Order Entry/Maintenance window
changing orders 2–47

Order filter section
Find Order window 2–45

Order Line section
SportsPro Order Tracking System 2–47

Order section
adding buttons 3–48
Navigation panel 2–40
SmartDataViewer 2–40

P

Page status
  AppBuilder status area 1–11

Pages dialog box 3–73

Partially assembled application
  viewing 3–62

Phone-in consultation 5–5

Pointer object
  AppBuilder Object Palette 1–12

Prev/Next spin box 3–56, 3–60

Print button
  AppBuilder toolbar 1–9

PRO*Tools
  defined 1–4

Procedure Editor
  defined 1–3

Procedure files 4–8

Procedure Properties button
  AppBuilder toolbar 1–9

Procedure Settings dialog box 3–72

Progress
  Consulting Services 5–4
  Documentation Set 5–3
  Education Services 5–3
  on-site training 5–3
  Technical Services 5–5
    Progress Knowledge Base 5–7
    Technical Support Centers 5–6

Progress Advisor 3–2

Progress training courses
  complete list 5–4

property 3–28

Property sheets
  CustNum field 4–33
  Image 3–12
  in AppBuilder 3–8
  main window frame 4–42
  rectangle 3–14
  SmartDataViewer 3–28
  SmartWindow 3–7, 3–20, 3–36
  text label 3–17

PSTimer object
  AppBuilder Object Palette 1–14

Publish and Subscribe module 2–24
  components 2–25

Q

Query Builder window 4–15

Query object
  AppBuilder Object Palette 1–12

Querying
  order information 3–39

R

Radio-Set object
  AppBuilder Object Palette 1–13

Rectangle object
  AppBuilder Object Palette 1–13

Rectangle property sheet 3–14

Rectangles
  adding to SmartWindow 3–13

Registering for classes online 5–4

Report Builder
  defined 1–4

Resizable Dynamic Browser module 2–17
  components 2–19

Results
  defined 1–4

Run button
AppBuilder toolbar 1–9
running sample application 2–7

Running tools
instances limit 1–3

S

Save button
AppBuilder toolbar 1–8

Search criteria 2–44

Search query
Find Order window 2–44

Section 3–37

Section Editor
adding trigger code 3–37
New button 3–56
saving new code 3–62

Selection of consulting services 5–4

Selection-List object
AppBuilder Object Palette 1–13

Set Application Language window 2–51

Slider object
AppBuilder Object Palette 1–13

SmartB2B and SonicMQ module 2–27
components 2–28

SmartB2BObject object
AppBuilder Object Palette 1–15

SmartBusinessObject object
AppBuilder Object Palette 1–15

SmartConsumer object
AppBuilder Object Palette 1–15

SmartContainer object
AppBuilder Object Palette 1–15

SmartContainers
definition 4–4

SmartDataBrowser object
AppBuilder Object Palette 1–14

SmartDataBrowser Wizard 4–37

SmartDataBrowsers 4–35
adding to SmartWindow 3–75
creating 4–36
Inventory Trans dialog box 2–50
Item Maintenance window 2–49
Item Selection window 2–41
resizing 4–42
saving 4–44
selecting fields 4–38
starting 4–36

SmartDataField module 2–22
components 2–23

SmartDataField object
AppBuilder Object Palette 1–14

SmartDataObject module 2–20
components 2–21

SmartDataObject object
AppBuilder Object Palette 1–14

SmartDataObject Wizard 4–14

SmartDataObjects
adding to Order Section 3–43
adding to SmartWindow 3–24
defining queries 4–14
fabricating 4–12
querying databases 4–14
saving 4–19
starting 4–13

SmartDataViewer object
AppBuilder Object Palette 1–14

SmartDataViewer Wizard 4–23

SmartDataViewers
adding field labels 4–31
adding to Customer Section 3–26
adding to SmartWindow 3–26
arranging field displays 4–29
changing field labels 4–31
changing layouts 4–29
creating 4–21
disabling fields 4–32
Display Customer window 2–36
Item Maintenance window 2–49
logical pages 2–47
Maintain Customer window 2–37
Order Entry/Maintenance window 2–41
Order Line information 2–47
Order Line section 2–47
Order section 2–40, 3–46
property sheet 3–28
saving 4–33
selecting table and fields 4–25
starting 4–22

SmartFilter
Find Order window 2–44

SmartFilter object
AppBuilder Object Palette 1–14

SmartFolder Attributes dialog box 3–69

SmartFolder object
AppBuilder Object Palette 1–14

SmartFolders 3–72
adding 3–75
adding SmartObjects to pages 3–75
adding to SmartWindow 3–66
Browse tab 2–48
choosing startup page 3–74
creating pages 3–72
Detail tab 2–48
instance attributes 3–69
logical pages 3–66

SmartLinks 3–2
creating links between 3–28
data 4–6
defined 4–4
navigation 4–6
page 4–6
types 4–5

SmartLinks dialog box 3–29

SmartObject applications 4–10

SmartObject object
AppBuilder Object Palette 1–14

SmartObjects
adding to SmartFolder page 3–75
controlling displays 3–66
defined 4–2
internal procedures 4–2
masters 4–11
running persistently 4–2
SmartContainers 4–4
SmartLinks 4–4
templates 4–2
used in main window 4–2

SmartPanel object
AppBuilder Object Palette 1–14

SmartPanels
adding to SmartFolder page 3–79

SmartPaste
advanced editing features 1–18

SmartProducer object
AppBuilder Object Palette 1–15

SmartReceiver object
AppBuilder Object Palette 1–15

SmartRouter object
AppBuilder Object Palette 1–15

SmartSelect Browser 2–41

SmartSender object
AppBuilder Object Palette 1–15

SmartToolBar object
AppBuilder Object Palette 1–14

SmartWindows
adding images 3–11
adding OCX controls 3–51
adding rectangles 3–13
adding SmartFolder 3–66
adding SmartObjects 3–22
adding spin boxes 3–51
adding text labels 3–16
advanced property sheet 3–8
assembling main window 3–3
basic property sheet 3–7
creating 3–5
property sheet 3–7, 3–20, 3–36
saving 3–19

Spin boxes
adding to SmartWindow 3–51
OCX control 3–51

Sports2000 database 2–2
connecting from Data Dictionary 2–2

SportsPro Order Tracking System 2–8
business scenario 4–9
components 4–8
customer maintenance components 2–38
Customer section 2–34
development phases 4–10
find order components 2–46
inventory transfer components 2–50
main window components 2–32
menu items 2–49
order entry/maintenance components 2–42
Order Line section 2–47
set application language components 2–52

SportsPro Sample Application
Drag and Drop module 2–15
Dynamic Buffer/Temp Table module 2–12
Dynamic Query module 2–10
main window components 2–8
Publish and Subscribe module 2–24
Resizable Dynamic Browser module 2–17
running completed application 2–6
SmartB2B and SonicMQ module 2–27
SmartDataField module 2–22
SmartDataObject module 2–20
using PRO*Tools to run 2–7

Starting a tool 1–2

Startup on Page dialog box 3–74

Status area
AppBuilder 1–11
logical page 3–74

Stop button
AppBuilder toolbar 1–9

Syntax expansion
advanced editing features 1–17

Syntax indenting
advanced editing features 1–18

T

Tabs
in SmartFolders 2–47

Technical support 5–6

Text field
AppBuilder 1–10

Text labels
adding to SmartWindows 3–16
selecting colors 3–17

Text object
AppBuilder Object Palette 1–13

Third-party objects
OCS Controls 3–50

Title field

Toggle-Box object
AppBuilder Object Palette 1–13

Toolbar
Desktop 1–2

Tools
Actuate e.Report Designer Professional 1–5
AppBuilder 1–4
Application Compiler 1–5
Application Debugger 1–5
Data Administration 1–3
Data Dictionary 1–3
launching from ADE Desktop 1–2
launching from another tool 1–3
PRO*Tools 1–4
Procedure Editor 1–3
Report Builder 1–4
Results 1–4
Translation Manager 1–4
Visual Translator 1–4
Web Tools 1–4

Tools menu
AppBuilder 1–7
list of tools 1–3
starting another tool 1–3

Toolset
list of tools 1–3

ToolTips 1–2
adding to buttons 3–33

Training classes 5–3

Translation Manager
defined 1–4

Trigger code
adding to buttons 3–37
adding to spin box 3–56
adding with Section Editor 3–37

Typographical conventions 1–xi

U

Update panel 2–40

Update SmartPanel
in Item Maintenance window 2–49

User-interface triggers 3–61
fetching first record 3–61
fetching last record 3–61
fetching next record 3–60
fetching previous record 3–56

V

Viewing order information 3–39

Visual Translator
defined 1–4

W

Web Tools
defined 1–4

Window files 4–8

Window menu
AppBuilder 1–8

Wizards
SmartDataBrowser 4–35, 4–37
SmartDataObject 4–12
SmartDataViewer 4–21

Working directory 2–2